1. Print all the phone numbers.

awk is a tool that can be used to filter data

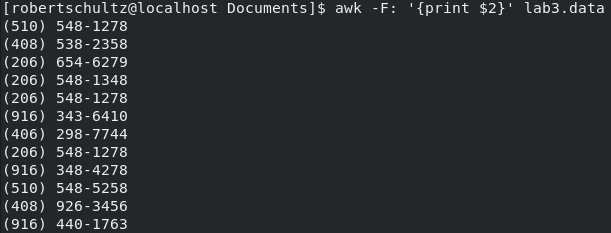
it is also pretty easy to use

With '-F:' we tell the computer to separate the data by ':'.

Normally, awk uses white space as a separator, but this option changes that.

Using '{print $2}' we are telling the computer to print what occurs after the first separator, but before the second.

awk -F: '{print $2}' lab3.data



2. Print Dan's phone number.

Same as 1, but this time we tell the computer to look for lines that contain 'Dan'.

awk -F: '/Dan/{print $2}' lab3.data

Macintosh HD:Users:robertschultz:Desktop:awk:awk_1_02.png

3. Print Susan's name and phone number.

Same as 2, but now we are looking for 'Susan'.

Also, we tell the computer to print what comes before and after the first separator but only before the second.

If my description is confusing, sorry, I'm just having trouble putting this into words.

awk -F: '/Susan/{print $1,$2}' lab3.data

Macintosh HD:Users:robertschultz:Desktop:awk:awk_1_03.png

4. Print all last names beginning with S.

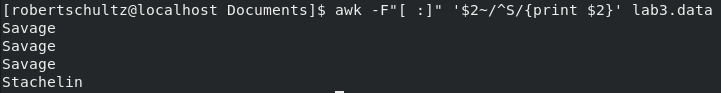
This time, we change the separator to both a space and a ':'.

This separates the first name from the last name, and the last name from the rest of the data.

Using '^S' we tell the computer we are looking for words that start with an 'S'.

'$2~' tells the computer to look at the second word, without that it would look for lines that started with an 'S'.

awk -F"[ :]" '$2~/^S/{print $2}' lab3.data



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

We don't need to change the separator this time.

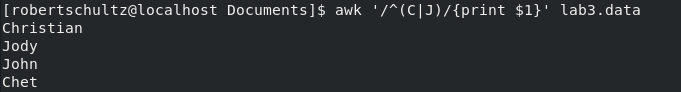
Using '^' tells the computer to look at the start of each line.

(C|J) tells the computer we want the line to start with either a 'C' or a 'J'.

Using '{print $1}' we tell the computer to print only the first word on each line that matches our criteria.

Ok, I think I understand how to put this into words now.

awk '/^(C|J)/{print $1}' lab3.data



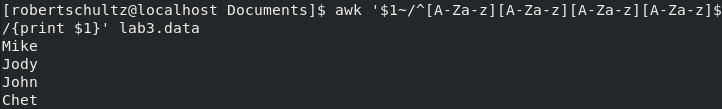
6. Print all first names containing only four characters.

Since we want to check the first word, we use '$1~'. This tells the computer to try to find matches for our criteria on the first word of each line only.

'{print $1}' tells the print only the first word.

Having both '^' and '$' to designate endpoints allows us to have a specific amount of characters. If either of those were absent, anything with at least four characters would meet the criteria.

awk '$1~/^[A-Za-z][A-Za-z][A-Za-z][A-Za-z]$/{print $1}' lab3.data

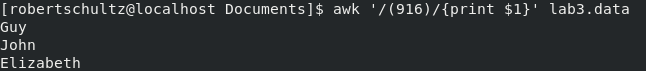


7. Print the first names of all those in the 916 area code.

All area codes in the data are in parentheses, we can use those to find only instances of 916 that are in the space for area code.

If we used '-F"[ :]"', then '$3~/916/{print $1}' would work as well.

awk '/(916)/{print $1}' lab3.data



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

Again we have to change the separator. '-F:' changes the separator to ':'.

Using '"$"' tells the computer to print that character as well. This places the dollar sign before each dollar amount.

I think I've already explained how changing separators work enough times.

awk -F: '/Mike/{print "$"$3,"$"$4,"$"$5}' lab3.data

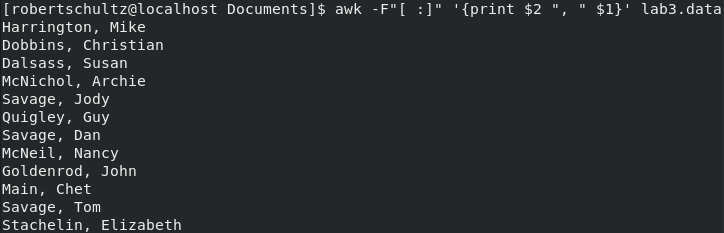
Macintosh HD:Users:robertschultz:Desktop:awk:awk_1_08.png

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

I won't explain everything in detail again. We changed the separator to both space and ':'.

By adding '", "' we tell the computer to print a comma and a space between the last name and first name.

awk -F"[ :]" '{print $2 ", " $1}' lab3.data



10. Write an awk script called facts that

Prints full names and phone numbers for the Savages.

There is probably a better way to do this.

Since we are already in the script, using '-F:' will not work. Using 'F"[:]"' will have the same effect that option would have.

As an added benefit, we only need to say use that one time. The change of separator carries over to the rest of the script.

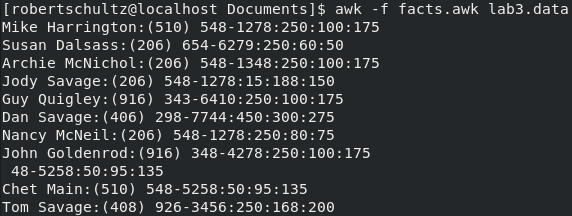
F"[:]" $1~/Savage/ {print $1,$2}

Prints Chet's contributions.

$1~/Chet/ {print $3,$4,$5}

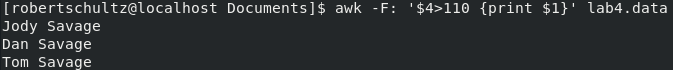
Prints all those who contributed $50 the first month.

$3~/50/{print $0}



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

awk -F: '$4>110 {print $1}' lab4.data



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

I've already explained just about everything that was needed here.

'$5<75 is the only thing I haven't explained. The '<' tells the computer what I'm looking for is cases where the value found in the data is less than the value I've specified.

awk -F: '$5<75 {print $1,$2}' lab4.data

Macintosh HD:Users:robertschultz:Desktop:awk:awk_2_02.png

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

Was convinced I was doing something wrong until I actually looked at the data.

Using both '<' and '>' I define a range. '&&' makes the computer look for cases where both statements are true.

When I did not get any results, I decided to look at the data. It turns out there are no cases where both statements are true, so nothing falls within that range.

awk -F: '$3<150 && $3>75 {print $1}' lab4.data

Macintosh HD:Users:robertschultz:Desktop:awk:awk_2_03.png

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

Using 'sum=' I can set the value of 'sum' to be the total of any number of values.

Then I use '>' to check if 'sum' is greater than 700.

awk -F: '{sum=$3+$4+$5} sum>700{print $1}' lab4.data

Macintosh HD:Users:robertschultz:Desktop:awk:awk_2_04.png

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

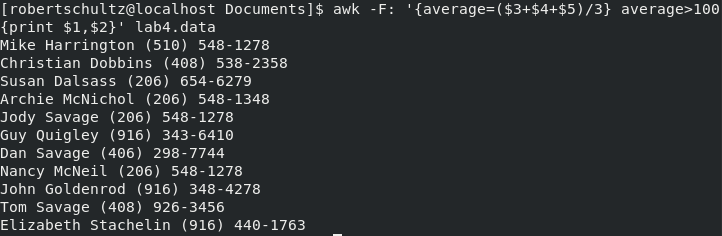
This time I used 'average='. I basically use it the same way I would use 'sum'.

However, I needed to ensure that division occurred only at the end.

Following order of operations, division comes first. Putting the first part in parentheses tells the computer that part comes first instead.

'>' was used again to check that 'average' was greater than 100.

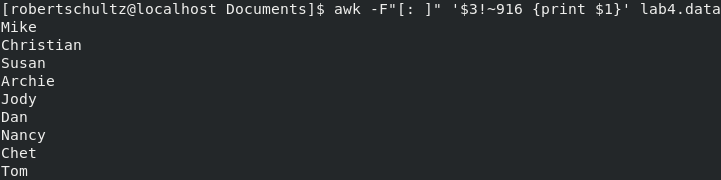
awk -F: '{average=($3+$4+$5)/3} average>100{print $1,$2}' lab4.data



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

'!~' signals that I am looking for a value that does not match.

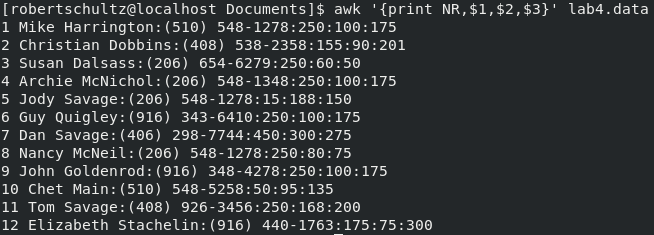
awk -F"[: ]" '$3!~916 {print $1}' lab4.data



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

'NR' is a special variable that refers to the record number.

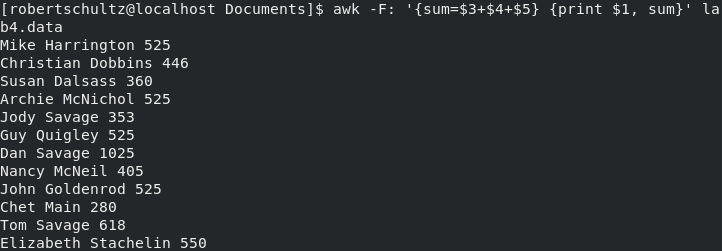
awk '{print NR,$1,$2,$3}' lab4.data



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

Using 'sum' again, but this time we are print the value instead of checking it.

awk -F: '{sum=$3+$4+$5} {print $1, sum}' lab4.data



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

We just need to print out the value of Chet's second contribution. However, we add '10' to that value.

awk -F: '$1~/Chet/{print $4+10}' lab4.data

Macintosh HD:Users:robertschultz:Desktop:awk:awk_2_09.png

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

Using 'gsub' we can identify a string we want to change.

We first define the target as 'Nancy Mcneil' and our replacement as 'Doris Shutt'.

awk -F: '{gsub(/Nancy Mcneil/,"Doris Shutt")}' lab4.data

Macintosh HD:Users:robertschultz:Desktop:awk:awk_2_10.png

Most of what I used was found in Awk - A Tutorial and Introduction - by Bruce Barnett