

Lindsay Graham

Linux Administration

Week 4 Lab – Awk

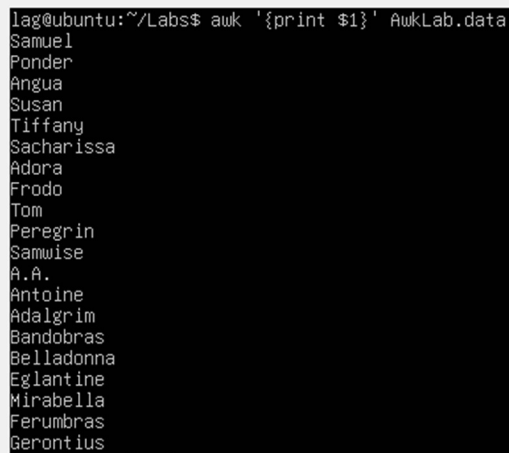
Awk Tutorial

Awk is a powerful scripting language that is ideal for tasks like pattern matching and data manipulation. Awk commands can be run through scripts or directly at the command line. Awk excels at processing text files: it scans lines, separates data based on delimiters (whitespace by default, or user-defined), compares lines to patterns, and allows you to perform various actions like printing or formatting the results. Throughout this tutorial, the file that was used was named AwkLab.data. For clarity, it will be referred to as 'filename' throughout the tutorial.

Example 1. Print all the First Names

- **awk '{print \$1}' filename**

Figure 1.1 Output showing all the First Names in the file



```
lag@ubuntu:~/Labs$ awk '{print $1}' AwkLab.data
Samuel
Ponder
Angua
Susan
Tiffany
Sacharissa
Adora
Frodo
Tom
Peregrin
Samwise
A.A.
Antoine
Adalgrim
Bandobras
Belladonna
Eglantine
Mirabella
Ferumbas
Gerontius
```

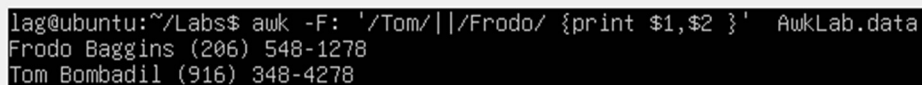
The syntax for this command is as follows: `awk '{print $1}' filename`.

Awk separates lines by whitespace by default and uses built-in variables (\$1,\$2,\$3, etc.) for each field (or section) of each line in the file. So, this command essentially instructs awk to print the first word (\$1) of every line in the specified file.

Example 2. Print phone numbers for Tom and Frodo after their names

- **awk -F: '/Tom/|/Frodo/ {print \$1,\$2}' filename**

Figure 2.1 Output showing Tom and Frodo's names and phone numbers



```
lag@ubuntu:~/Labs$ awk -F: '/Tom/|/Frodo/ {print $1,$2}' AwkLab.data
Frodo Baggins (206) 548-1278
Tom Bombadil (916) 348-4278
```

The syntax for this command is as follows: `awk -F: '/Tom/|/Frodo / {print $1,$2}' filename`.

In this example, the -F option tells awk to use a colon (:) as the delimiter instead of the default whitespace. The patterns to match (i.e., "Tom" and "Frodo") are separated by two pipes (||). The pipes represent that awk will match lines containing either "Tom" or "Frodo" or both. If a match is found, the print command prints the first and second fields (\$1, \$2) of the line, separated by a space – in this example, a name and phone number.

Example 3. Print Peregrin's full name and phone number area code only

- `awk '{ if ($1=="Peregrin") {print $1,$2}}' filename`

Figure 3.1 Output of Peregrin's full name and phone number area code

```
lag@ubuntu:~/Labs$ awk '{ if ($1=="Peregrin") {print $1,$2}}' AwkLab.data
Peregrin Took:(510)
```

The syntax for this command is as follows: `awk '{if ($1=="Peregrin") {print $1,$2}}' filename`.

This command uses an if condition to say that if the first field (\$1) of a line equals the pattern to match ("Peregrin"), then print the first and second fields of the line which contain the name and phone number area code.

Example 4. Print all phone numbers (full number) in the 123 area code along with the names

- `awk -F: '/123/ {print $1,$2}' filename`

Figure 4.1 Output showing all phone numbers with the 123 area code along with names

```
lag@ubuntu:~/Labs$ awk -F: '/123/ {print $1,$2}' AwkLab.data
Antoine de Saint-Exupery (123) 978-6432
Belladonna Took (123) 978-5754
Eglantine Took (123) 978-3574
```

The syntax for this command is as follows: `awk -F: '/123/ {print $1,$2}' filename`.

In this example, the command instructs awk to use ":" as a delimiter and find all lines containing '123'. The print command then says to print the first and second fields only using the new delimiter.

Example 5. Print all Last names beginning with either a T or D

- `awk 'split($2,a,":") split($3,b,":") {if ($2 ~ /^[TD]/) {print a[1]}} {if ($3 ~ /^[TD]/) {print b[1]}}' filename`

Figure 5.1 Output of last names that begin with a T or D

```
lag@ubuntu:~/Labs$ awk 'split($2,a,":") split($3,b,":") {if ($2 ~ /^[TD]/) {print a[1]}} {if ($3 ~ /^[TD]/) {print b[1]}}' AwkLab.data
Dearheart
Took
Took
Took
Took
Took
Took
Took
```

The syntax for this command is as follows: `awk 'split($2,a,":") split($3,b,":") {if ($2 ~ /^[TD]/) {print a[1]}} {if ($3 ~ /^[TD]/) {print b[1]}}' filename`.

The `split()` command can split a field (or section) into an array based on a delimiter. In the sample file, last names were present in both the second (\$2) and third (\$3) fields. Additionally, each field containing a last name was followed by a colon and an area code. This command splits the second field into the “a” array and the third field into the b array, using a colon as the delimiter. Then, it uses regular expressions to check if the second field starts with a "T" or "D" (signified by the caret ^). If it does, the first element of the “a” array (a[1]) containing the last name is printed. Similarly, if the third field starts with a "T" or "D", the first element of the b array (b[1]) is printed.

Example 6. Print all first names containing four or less characters

- `awk '{if (length($1) <= 4) {print $1}}' filename`

File 6.1 Output showing all first names containing four or less characters

```
lag@ubuntu:~/Labs$ awk '{if (length($1) <= 4) {print $1}}' AwkLab.data
Tom
A.A.
```

The command syntax is as follows: `awk '{if (length($1) <=4) {print $1}}' filename`.

The if statement, when used with awk, allows you to specify a condition and an action to be performed if the condition is true. In this example, the first field holds the first names. If a name is less than 4 characters long, it will be printed.

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

- `awk 'split($2, a, ":") {if (a[2]~/ (916)/) {print $1,a[2]} split($3, b, ":") {if (b[2]~/ (916)/) {print $1,b[2]}}' filename`

Figure 7.1 Result showing the first names and area codes of all those in the 916 area code

```
lag@ubuntu:~/Labs$ awk 'split($2, a, ":") {if (a[2]~/ (916)/) {print $1,a[2]} split($3, b, ":") {if (b[2]~/ (916)/) {print $1,b[2]}}' AwkLab.data
Sacharissa (916)
Tom (916)
A.A. (916)
```

The command syntax is as follows: `awk 'split($2, a, ":") {if (a[2]~/916/) {print $1,a[2]} split($3, b, ":") {if (b[2]~/916/) {print $1,b[2]}}' filename`.

As mentioned earlier, the `split()` command can split a field into an array based on a delimiter. In the sample file, area codes are held in fields \$2 and \$3. The if statement checks if the second element of the a or b array (which contains the area code) matches "(916)". If it does, the first field (containing the last name) is printed along with the area code from the corresponding array element (either a[2] or b[2]).

Example 8. Print Sacharissa’s campaign contributions following her name. Each value should be printed with a leading dollar sign, e.g., \$250 \$100 \$175

- **awk -F ":" '/^Sacharissa/ {print \$1, "\$"\$3, "\$"\$4, "\$"\$5}' filename**

Figure 8.1 Results showing Sacharissa's campaign contributions following her name with a leading dollar sign.

```
lag@ubuntu:~/Labs$ awk -F ":" '/^Sacharissa/ {print $1, "$"$3, "$"$4, "$"$5}' AwkLab.data
Sacharissa Cripslock $250 $100 $175
lag@ubuntu:~/Labs$ _
```

The command syntax is as follows: `awk -F ":" '/^Sacharissa/ {print $1, "$"$3, "$"$4, "$"$5}' filename`.

In this example, the delimiter is set to a colon. Then, `awk` is used to search for a field that starts with "Sacharissa," which is a first name field found in `$1`. If a match is found, the first name field (`$1`) is printed along with the campaign contributions found in fields `$3`, `$4`, and `$5`. To include the leading dollar symbol, the symbol must be enclosed in double quotes in the print statement.

Example 9. Print last names followed by a comma and the phone number

- **awk -F ":" 'split(\$1,a," ") {print a[2]a[3]","\$2}' filename**

Figure 9.1 Sample of results showing last name followed by a comma and phone number

```
lag@ubuntu:~/Labs$ awk -F ":" 'split($1,a," ") {print a[2]a[3]","$2}' AwkLab.data
Vimes,(510) 548-1278
Stibbons,(408) 538-2358
vonÜberwald,(206) 654-6279
StoHelit,(206) 548-1348
Aching,(206) 548-1278
Cripslock,(916) 343-6410
BelleDearheart,(406) 298-7744
Baggins,(206) 548-1278
Bombadil,(916) 348-4278
Took,(510) 548-5258
Gamgee,(408) 926-3456
Milne,(916) 440-1763
deSaint-Exupery,(123) 978-6432
Took,(345) 978-7684
"Bullroarer"Took,(453) 978-3534
Took,(123) 978-5754
Took,(123) 978-3574
Took,(345) 978-2677
IIITook,(563) 978-753
Took,(574) 978-8535
lag@ubuntu:~/Labs$
```

The command syntax is as follows: `awk -F ":" 'split($1,a," ") {print a[2]a[3]","$2}' filename`.

In this example, the last names needed to be printed with a comma and phone number. First, the delimiter was set to a colon. Because the first colon is either in the second or third whitespace-separated section, splitting it allowed the names to be placed in an array where they could be printed. Positions `a[2]` and `a[3]` signified the fields that had the last names. They were printed along with field `$2`, which is the phone number, and the second field after the colon. To print the comma, it must be enclosed in double quotes within the print command.

Example 10. Print the first and last names of those who contributed more than \$110 in the last month. Include their last month contribution amount after the name.

- `awk -F ":" 'split($1,a," ") {if ($5>110) {print a[1]" "a[2]" "a[3], "$5}}' filename`

Figure 10.1 Screenshot of the first and last names of those who contributed more than \$110 with their last month contribution.

```
lag@ubuntu:~/Labs$ awk -F ":" 'split($1,a," ") {if ($5>110) {print a[1]" "a[2]" "a[3], "$5}}' AwkLab.data
Samuel Vimes $175
Ponder Stibbons $201
Susan Sto Helit $175
Tiffany Aching $150
Sacharissa Cripslock $175
Adora Belle Dearheart $275
Tom Bombadil $175
Peregrin Took $135
Samwise Gamgee $200
A.A. Milne $300
Antoine de Saint-Exupery $175
Adalgrim Took $467
Bandobras "Bullroarer" Took $4673
Belladonna Took $175
Eglantine Took $4367
Mirabella Took $175
Ferumbras III Took $3457
Gerontius Took $4562
```

The command syntax is as follows: `awk -F ":" 'split($1,a," ") {if ($5>110) {print a[1]" "a[2]" "a[3], "$5}}' filename`.

In this example, `awk` was used to find individuals who contributed more than \$110 in their last month's contribution. The delimiter was set to a colon to simplify working with the contributions, which are all separated by colons. Next, the first field was split into an array by whitespace, representing the first and last name fields. An if statement then checked the last contribution, held in `$5`. If `$5` was greater than 110, the name and contribution were printed.

Example 11. Print the last names, phone numbers, and first month contribution of those who contributed less than \$150 in the first month.

- `awk -F ":" 'split($1,a," ") {if ($3<150) {print a[1]" "a[2]" "a[3], "$3}}' filename`

Figure 11.1 Results showing the last names, phone numbers, and first month contribution of those who contributed less than \$150 in the first month

```
lag@ubuntu:~/Labs$ awk -F ":" 'split($1,a," ") {if ($3<150) {print a[1]" "a[2]" "a[3], "$3}}' AwkLab.data
Tiffany Aching $15
Peregrin Took $50
```

The command syntax is as follows: `awk -F ":" 'split($1,a," ") {if ($3<150) {print a[1]" "a[2]" "a[3], "$3}}' filename`.

In this example, `awk` was used to find individuals who contributed less than \$150 in the first month. Like in the last example, the delimiter was set to a colon to simplify working with the contributions. Next, the first field was split into an array by whitespace, representing the first and last name fields. An if statement checked the first contribution, held in `$3`. If `$3` was less than \$150, the name and contribution were printed.

Example 12. Print the first names and contribution of those who contributed between \$10 and \$200 in the first month.

- `awk -F ":" 'split($1,a,"") {if ($3>=10 && $3<=200) {print a[1],"$" $3}}' filename`

Figure 12.1 Results showing the first names and contribution of those who contributed between \$10 and \$200 in the first month.

```
lag@ubuntu:~/Labs$ awk -F ":" 'split($1,a,"") {if ($3>=10 && $3<=200) {print a[1],"$" $3}}' AwkLab.data
Ponder $155
Tiffany $15
Peregrin $50
A.A. $175
```

The command syntax is as follows: `awk -F ":" 'split($1,a,"") {if ($3>=10 && $3<=200) {print a[1],"$" $3}}' filename`.

In this example, `awk` was used to find the names and contributions of individuals who contributed between \$10 and \$200 in the first month. This example was structured similarly to the previous two examples, but had a more complex argument in the `if` statement. The `&&` operator was used to specify two conditions. If the first month contribution, held in `$3`, was less than or equal to \$200 and greater than or equal to \$10, the first name and contribution were printed.

Example 13. Print the first name, last names and total contributions of those who contributed less than \$700 over the three-month period

- `awk -F ":" 'split($1,a,"") (sum=$3+$4+$5) {if (sum<700) {print a[1],a[2],a[3], "$"sum}}' filename`

Figure 13.1 Sample of results showing the first name, last names, and total contributions of those who contributed less than \$700 over the three month period.

```
lag@ubuntu:~/Labs$ awk -F ":" 'split($1,a,"") (sum=$3+$4+$5) {if (sum<700) {print a[1],a[2],a[3], "$"sum}}' AwkLab.data
Samuel Vimes $525
Ponder Stibbons $446
Angua von Überwald $360
Susan Sto Helit $525
Tiffany Aching $353
Sacharissa Cripslock $525
Frodo Baggins $405
Tom Bombadil $525
Peregrin Took $280
Samwise Gamgee $618
A.A. Milne $550
Antoine de Saint-Exupery $525
```

The command syntax is as follows: `awk -F ":" 'split($1,a,"") (sum=$3+$4+$5) {if (sum<700) {print a[1],a[2],a[3], "$"sum}}' filename`.

In this example, `awk` was used to find the names and total contributions of individuals who contributed less than \$700 over the three-month period. In addition to setting the delimiter and splitting the first field like in other examples, this example demonstrated how to add the values of fields and save them in a variable (i.e., `sum`). The three months of contributions are stored in `$3 + $4 + $5`, and the statement `sum = $3 + $4 + $5` creates a variable to

store the value of that sum. This variable can then be used in an if statement to check whether the value is less than \$700. If it is, the names and total contributions are printed.

Example 14. Print the first names and first letter of the last name, and average contribution of those who had an average contribution of more than \$300

- `awk 'split($2,a,":") (split(a[1],b,"") split(a[2],c,"") split($0,d,":") (sum=d[3]+d[4]+d[5]) {if ((match(b[1],/^[[:upper:]]*/)) && ((sum/3)>300)) {print $1,b[1], "$"(sum/3)} else if ((match(b[2],/^[[:upper:]]*/)) && ((sum/3)>300)) {print $1,b[2], "$"(sum/3)}}' filename`

Figure 14.1 Results showing the first names and first letter of the last name, and average contribution of those who had an average contribution of more than \$300

```
lag@ubuntu:~/Labs$ awk 'split($2,a,":") split(a[1],b,"") split(a[2],c,"") split($0,d,":") (sum=d[3]+d[4]+d[5]) {if ((match(b[1],/^[[:upper:]]*/)) && ((sum/3)>300)) {print $1,b[1], "$"(sum/3)} else if ((match(b[2],/^[[:upper:]]*/)) && ((sum/3)>300)) {print $1,b[2], "$"(sum/3)}}' AwkLab.data
Adora B $341.667
Adalgrim T $1746.67
Bandoabras " $3931.33
Eglantine T $1771.67
Mirabella T $507
Ferumbas I $1269
Gerontius T $1925
```

The command syntax is as follows: `awk 'split($2,a,":") (split(a[1],b,"") split(a[2],c,"") split($0,d,":") (sum=d[3]+d[4]+d[5]) {if ((match(b[1],/^[[:upper:]]*/)) && ((sum/3)>300)) {print $1,b[1], "$"(sum/3)} else if ((match(b[2],/^[[:upper:]]*/)) && ((sum/3)>300)) {print $1,b[2], "$"(sum/3)}}' filename`

In this example, awk was used to show the first names and the first letter of the last name, along with the average contribution of individuals who had an average contribution of more than \$300. Similar to the last example, the total contributions were saved in a variable. To calculate the average, this total was then divided by the number of contributions (3 in this case). The average was included in an if statement to check if it was more than \$300. The second field was split by a colon and then additional if statements checked to see if the first and second positions of the split field started with a capital letter using regular expressions (i.e., `[[:upper:]]`). If it was, the first name, the first letter of the last name, and the average contribution were printed.

*Please note that the regex did not work as intended to find only capital letters. I was unable to exclude the quote that surrounded a nick name.

Example 15. Print the last name and area code of those not in the 916 area code.

- `awk -F": " 'split($2,a," ") split($1,b," ") {if (a[1]!="(916)") {print b[2], b[3],$2}}' filename`

Figure 15.1 Results showing the last name and area code of those not in the 916 area code.

```
lag@ubuntu:~/Labs$ awk -F":" 'split($2,a," ") split($1,b," ") {if(a[1]!="(916)") {print b[2],b[3],$2}}' AwkLab.data
Vimes (510) 548-1278
Stibbons (408) 538-2358
von Überwald (206) 654-6279
Sto Helit (206) 548-1348
Aching (206) 548-1278
Belle Dearheart (406) 298-7744
Baggins (206) 548-1278
Took (510) 548-5258
Gamgee (408) 926-3456
de Saint-Exupery (123) 978-6432
Took (345) 978-7684
"Bullroarer" Took (453) 978-3534
Took (123) 978-5754
Took (123) 978-3574
Took (345) 978-2677
III Took (563) 978-753
Took (574) 978-8535
```

The command syntax is as follows: `awk -F":" 'split($2,a," ") split($1,b," ") {if(a[1]!="(916)") {print b[2], b[3],$2}}' filename`

In this example, `awk` was used to show the last name and area code of individuals who were not in the 916 area code. First, the delimiter was set to a colon, and then the `$2` field (containing the phone number) was split. Then the first field was split by a space. An if statement checked to see if the first position of the split `$2` field, which represented the area code, was not equal to 916 (signified by `!=`). If true, the last name (from the split first field) and phone number were printed.

*Please note that I am still struggling how to capture the last name when there are middle names and last names in two different positions.

Example 16. Print each record preceded by the number of the record.

- `awk '{print NR" ",$0}' filename`

Figure 16.1 Results showing each record preceded by the number of the record.

```
lag@ubuntu:~/Labs$ awk '{print NR" ",$0}' AwkLab.data
1 Samuel Vimes:(510) 548-1278:250:100:175
2 Ponder Stibbons:(408) 538-2358:155:90:201
3 Angua von Überwald:(206) 654-6279:250:60:50
4 Susan Sto Helit:(206) 548-1348:250:100:175
5 Tiffany Aching:(206) 548-1278:15:188:150
6 Sacharissa Cripslock:(916) 343-6410:250:100:175
7 Adora Belle Dearheart:(406) 298-7744:450:300:275
8 Frodo Baggins:(206) 548-1278:250:80:75
9 Tom Bombadil:(916) 348-4278:250:100:175
10 Peregrin Took:(510) 548-5258:50:95:135
11 Samwise Gamgee:(408) 926-3456:250:168:200
12 A.A. Milne:(916) 440-1763:175:75:300
13 Antoine de Saint-Exupery:(123) 978-6432:250:100:175
14 Adalgrim Took:(345) 978-7684:4673:100:467
15 Bandobras "Bullroarer" Took:(453) 978-3534:6753:368:4673
16 Belladonna Took:(123) 978-5754:356:247:175
17 Eglantine Took:(123) 978-3574:473:475:4367
18 Mirabella Took:(345) 978-2677:783:563:175
19 Ferumbas III Took:(563) 978-753:250:100:3457
20 Gerontius Took:(574) 978-8535:535:678:4562
```


In this example, awk was used to show each record preceded by its number. The command syntax is as follows: `awk '{print NR" "$0}' filename`. The NR variable keeps count of all the lines in the file. The \$0 variable stands for a whole line. Printing the NR and \$0 variables will give you a numbered list of each line along with the line itself.

Example 17. Print the name and total contribution of each person

- `awk -F":" 'split($1,a," ") (sum=$3+$4+$5) {print a[1],a[2],a[3], "$"sum}' filename`

Figure 17.1 Results showing the name and total contribution of each person

```
lag@ubuntu:~/Labs$ awk -F":" 'split($1,a," ") (sum=$3+$4+$5) {print a[1],a[2],a[3], "$"sum}' AwkLab.data
Samuel Vimes $525
Ponder Stibbons $446
Angua von Überwald $360
Susan Sto Helit $525
Tiffany Aching $353
Sacharissa Cripslock $525
Adora Belle Dearheart $1025
Frodo Baggins $405
Tom Bombadil $525
Peregrin Took $280
Samwise Gamgee $618
A.A. Milne $550
Antoine de Saint-Exupery $525
Adalgrim Took $5240
Bandobras "Bullroarer" Took $11794
Belladonna Took $778
Eglantine Took $5315
Mirabella Took $1521
Ferumbas III Took $3807
Gerontius Took $5775
```

The command syntax is as follows: `awk -F":" 'split($1,a," ") (sum=$3+$4+$5) {print a[1],a[2],a[3], "$"sum}' filename`.

In this example, awk was used to print the name and total contribution of each person. Like in prior examples, the first field was split by a space and the whole line was split by a colon. The sum of all the contributions was stored in a variable sum and printed.

Example 18. Add \$10 to Tiffany Aching's first contribution and print her full name and first contribution.

- `awk -F":" 'split($1,a," ") {if (a[1]=="Tiffany") {print a[1],a[2], "first contribution: $"$3, "plus $10: $"($3+10)}}' filename`

Figure 18.1 Results showing Tiffany Aching's full name, first contribution, first contribution + \$10

```
lag@ubuntu:~/Labs$ awk -F":" 'split($1,a," ") {if (a[1]=="Tiffany") {print a[1],a[2], "first contribution: $"$3, "plus $10: $"($3+10)}}' AwkLab.data
Tiffany Aching first contribution: $15 plus $10: $25
```

The command syntax is as follows: `awk -F":" 'split($1,a," ") {if (a[1]=="Tiffany") {print a[1],a[2], "first contribution: $"$3, "plus $10: $"($3+10)}}' filename`

In this example, awk was used to find Tiffany Aching's name and first contribution and then add \$10 to it. An if statement told awk to find a record that matched "Tiffany" and then print the first contribution, stored in \$3. Then 10 was added to \$3 to create the new gift. Text was added to explain which contribution was which.

Example 19. Change Samwise Gamgee's name to Sean Astin

- `awk -F" " '{gsub(/Samwise/, "Sean", $1) split($2, a, ":") gsub(/Gamgee/, "Austin", a[1])} {if ($1~/^Sean/) {print $1, a[1]}}' filename`

Figure 19.1 Results showing Samwise Gamgee's name changed to Sean Astin

```
lag@ubuntu:~/Labs$ awk -F" " '{gsub(/Samwise/, "Sean", $1) split($2, a, ":") gsub(/Gamgee/, "Austin", a[1])} {if ($1~/^Sean/) {print $1, a[1]}}' AwkLab.data
Sean Austin
```

In this example, awk was used with gsub to find the name "Samwise" and substitute it with "Sean". In this example, the gsub syntax instructs awk to find a pattern, then lists the text to substitute in quotes, and the location to look and place the new information. The same gsub was repeated to replace "Samwise Gamgee"'s last name with "Austin". Then an if statement searched for the new first name and printed the results.

Example 20. Write an awk script to do the following:

- (A.) Prints first name of the all the Tooks followed by their total campaign contributions .
- (B.) Print the full names and contributions of anyone who contributed between \$10 and \$200 in the last contribution
- (C.) Prints the full names and average contribution of those who contributed less than \$300 on average

Figure 20.1 Awk script in Linux

```
GNU nano 7.2                                awkscript.awk
#!/usr/bin/awk -f
{
    FS=":"

    {split($1, a, " ")

    ##First names and total contributions for the Tooks
    #Section 1
    if (a[2]=="Took" || a[3]=="Took")
    {print "A.) " a[1], " $"($3+$4+$5)};

    ##Full names and contributions of anyone who contributed between $10 and $200 in the last contribution
    #Section 2
    if ($5>=10 && $5<=200){
    {print "B.)" $1, " $"$5};
    }

    ##Full names and average contribution of those who contributed less than $300 on average
    #Section 3
    (tot=$3+$4+$5)
    (avg=(tot/3))
    if (avg<300){
    {print "C.)" $1, "avg:", "$"avg};
    }
    }
}
```

Figure 20.2 Results of ./awkscript.awk AwkLab.data

```

lag@ubuntu:~/Labs$ ./awkscript.awk AwkLab.data
C.)Samuel avg: $182.667
C.)Ponder Stibbons avg: $148.667
B.)Angua von Überwald $50
C.)Angua von Überwald avg: $120
B.)Susan Sto Helit $175
C.)Susan Sto Helit avg: $175
B.)Tiffany Aching $150
C.)Tiffany Aching avg: $117.667
B.)Sacharissa Cripslock $175
C.)Sacharissa Cripslock avg: $175
B.)Frodo Baggins $75
C.)Frodo Baggins avg: $135
B.)Tom Bombadil $175
C.)Tom Bombadil avg: $175
A.) Peregrin $280
B.)Peregrin Took $135
C.)Peregrin Took avg: $93.3333
B.)Samwise Gamgee $200
C.)Samwise Gamgee avg: $206
C.)A.A. Milne avg: $183.333
B.)Antoine de Saint-Exupery $175
C.)Antoine de Saint-Exupery avg: $175
A.) Adalgrim $5240
A.) Bandobras $11794
A.) Belladonna $778
B.)Belladonna Took $175
C.)Belladonna Took avg: $259.333
A.) Eglantine $5315
A.) Mirabella $1521
B.)Mirabella Took $175
A.) Ferumbras $3807
A.) Gerontius $5775
lag@ubuntu:~/Labs$ _

```

In this example the awk script: prints first name of the all the Tooks followed by their total campaign contributions (A.), the full names and contributions of anyone who contributed between \$10 and \$200 in the last contribution (B.), and the full names and average contribution of those who contributed less than \$300 on average (C.). The output is shown with a letter to relate it to the specific objective A,B, and C. I was unable to find a way to organize the output with headers into category where the results could be easily matched to the command.

References

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

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

<https://linuxhandbook.com/awk-if-else/>

<https://stackoverflow.com/questions/64143011/searching-for-multiple-strings-in-awk>

<https://www.unix.com/shell-programming-and-scripting/256168-awk-how-print-specific-field-if-string-matched.html>

<https://www.gnu.org/software/gawk/manual/gawk.html#toc-Regular-Expressions>

<https://ioflood.com/blog/awk-split/>