Simple Filtering using 'awk'

Lets figure out the syntax for awk

Filtering records containing the pattern "director"

```
awk -F"|" '/director/' sample.lst
```

-- is same as --

```
awk -F"|" '/director/ {print }' sample.lst
```

(Thus default action is print)

-- is same as --

```
awk -F"|" '/director/ {print $0}' sample.lst
```

($0 represents the whole line)

Syntax for awk

Thus it can be deduced that the syntax for awk is :

```
awk [options] 'pattern {action}' file(s)
```

- Searches for a pattern and applies action on it.
- Default action is to print current record on STDOUT.
- Default pattern is to match all lines.
- If file(s) not specified, input is taken from STDIN.

Common Options :

-F sets Field Separator (FS). Default is “ ” (space)

-f to read program/pattern from a file

```
echo “get me a cup of cofee” | awk '/coff?ee/'
```

We can use regular expressions that we generally use with grep (EREs)
(i.e. all egrep variants can be used here)

Splitting Fields using awk

```
echo "What's|up|doc" | awk -F"|" '{print $1,$2,$3}'
```

$1 represents the first field, $2 second … $n nth field.

$0 represents the whole line.

Can skip certain fields :

```
echo "Freedom|is|of|what|Speech" | awk -F"|" ' {print
$1,$3,$5}'
```

Can change value of fields using the same :

```
echo "APS Rocks" | awk '{$1="Scripting"; print}'
```

Formatting Output in awk

Can use the syntax of C’s printf to format the output

General Syntax : ‘`printf(%(format_specifier))`’

`Format Specifiers : d - Integers, s - Strings etc.`

Example :

```
awk -F "|" ' /[aA]gg?[ar]+wal/ { printf( "%s %s
%d\n\n",$2,$3,$6) } ' sample.lst
```

(Try out the other variants for it :)

Built - In Variables for awk

FS

Specifies Field Separator

OFS

Specifies Output Field Separator

RS

Specifies Record Separator

ORS

Specifies Output Separator

NF

Fields count of the line being processed

NR

Retrieves total count of records processed

Examples follow ahead :

Examples of Built - In Variables

Simulate Sed Behaviour : `'sed -n 5,10p sample.lst'`

```
awk ' NR==5,NR==10 {print} ' sample.lst
```

Printing Line Numbers :

```
awk -F "|" ' /[aA]gg?[ar]+wal/ { print NR,$2,$3,$6 } '
sample.lst
```

NF Variable always specifies the last field :

```
awk -F "|" '{OFS=":"} {print $2,$NF}' sample.lst
```

(Notice usage of OFS here)

(Try other variables yourself...)

Awk Pre Processing

Can use ‘BEGIN’ keyword to do something before awk processes data.

Optional Section

Can be used to generate report header, initialize variables, etc

Example :

```
awk -F "|" ' BEGIN{print "Report of Employees"} (NR<=10){
print NR,$2,$3,$6 } ' sample.lst
```

Awk Post Processing

Can use ‘END’ keyword to do something after awk is done processing data.

Also an optional Section

Can be used to print final result of computation, print output status, etc

Example :

```
awk -F "|" ' BEGIN{print "\nReport of Employees\n"} (NR<=10){
print NR,$2,$3,$6 } END{print "\nEnd of Report\n"}' sample.lst
```

Using shell commands in awk

Can use redirections and pipes.

Example :

```
awk -F "|" '{ printf( "%d %s %s %d\n\n",NR,$2,$3,$6) >
"output" } ' sample.lst
```

```
awk -F "|" '{ printf( " %d %s %s %d\n",NR,$2,$3,$6) |
"tr [a-z] [A-Z]" } ' sample.lst
```

(All the redirections and commands should be in double quotes only.)

Operators in awk

Relational Operators :

<, <=	Less, Less or Equal
>, >=	Greater, Greater or Equal
==, !=	Equal to, Not Equal to
~, !~	For Regex Comparison (Similar to)

Logical Operators :

&&	AND
||	OR
!	NOT

Operators in Action

Let’s say hello to the directors and chairmen in our data :

```
awk -F"|" '$3 == "director" || $3 == "chairman" {printf
"Hello %s %s %d \n",$2,$3,$6 } ' sample.lst
```

Operators in Action

Let’s say hello to the directors and chairmen in our data :

```
awk -F"|" '$3 == "director" || $3 == "chairman" {printf
"Hello %s %s %d \n",$2,$3,$6 } ' sample.lst
```

But this returns nothing!

Operators in Action

In the data, director is stored as “ director ”. Thus it won’t match $3=="director".

~ stands for regex (similar to). It finds/matches the pattern and doesn’t check exact equality.

```
awk -F"|" '$3 ~ "director" || $3 ~ "chairman" {printf
"Hello %s %s %d \n",$2,$3,$6 } ' sample.lst
```

Operators in Action

Lets print all salaries greater than 6000 :

```
awk '$NF >= 6000 {print}' sample.lst
```

Number Processing in awk

Arithmetic Operators :

+	Addition
-	Subtraction
*	Multiplication
/	Division
^	Exponentiation
%	Modulo

You can also create your own variables.

Number Processing in Action

Examples :

Simple Calculations :

```
echo 2 8 | awk '{print $1 ^ $2}'
```

Count number of people with salary at least 6000 :

```
    awk -F"|" 'BEGIN{count=0} $NF >= 6000 { count =
count + 1 ;} END{print count;} '
```

‘IF’ in awk

Awk supports ‘if’ conditional statement.

Example :

```
awk '{if ($1 > 30) print $1}' testfile
```

Use braces if you want to run multiple statements

```
awk '{if ($1 > 30)

{x = $1 * 3

print x}

}' testfile
```

‘IF...Else’ in awk

You can use if...else constructs

Example :

```
awk '{if ($1 > 30) {x = $1 * 3

print x} else {x = $1 / 2

print x}}' testfile
```

‘While’ in awk

Can use while loop to iterate over data with a condition

```
awk '{

sum = 0

i = 1

while (i < 5) {sum += $i

i++}

average = sum / 3

print "Average:",average}' testfile
```

Loop Control

Loops also support the usage of “break” and “continue” statements to control the iterations.

Try it yourself with some examples. (Explore :-))

Built in Functions in awk

Awk has a lot of built-in functions :

Math Functions like :

```
sin(x) | cos(x) | sqrt(x) | exp(x) | log(x) | rand()
```

String Functions like :

```
toupper | substr | split | index | length | & many more…
```

Refer to the man page of awk to find out about these.

(No need to remember them all)

```
$ man awk
```

Awk programs in a file

The ‘-f’ option lets you load an awk program from a file :

```
awk -f awk_prog.txt sample.lst
```

That's all Folks!
Any Doubts?

Pro Tips

Explore the commands and other combinations yourself too.

Check out the references! :-)

Ask your TAs.

References

- https://likegeeks.com/awk-command/
- https://pastebin.com/kHhPV5K0
- https://www.tutorialspoint.com/awk/