AWK - Part 2 Cheat Sheet

by Nouha_Thabet via cheatography.com/103894/cs/21387/

Print lines that contain a specific field

Let take the file awk.txt as an example:

```
AWK is awesome
Let learn AWK together
AWK is used for text processing
```

We want to print lines that contain the word "is":

```
awk '/is/ {print $0}'
 >>> AWK is awesome
 >>> AWK is used for text processing
```

What if we want to specify more than one field and print lines that contain the one of the these fields?
Let choose the fields "is" and "text", and let print "IS" before the line that contains the field "is" and print "TEXT" before the line that contains the field "text". Let see the example:

```
awk '/is/ {print " IS: ", $0} /text/ {print " TEX T:", $0}' awk.txt
 >>> IS: AWK is awesome
 >>> IS: AWK is used for text processing
 >>> TEXT: AWK is used for text processing
```

Here we see that the last line was printed twice because it contains the two fields.

awk -f

The f flag specifies that the following argument is the name of a file that contains an AWK program. Let use the file "command" as an example:

```
cat command
 >>> {print $0}
```

We use f flag to tell AWK to execute this pogram on awk.txt and we get the following result:

```
awk -f command awk.txt
 >>> AWK is awesome
 >>> Let learn AWK together
 >>> AWK is used for text processing
```

awk -F

The -F flag tell AWK to use the indicated argument as a field seperator. For example if we have fields separated by comma, then we need to specify the field separator.

awk -F (cont)

```
awk -F , '{print $2}'
 Yellow ,Bl ue,Red
 >>> Blue
```

We use `t` for tab separator.

In this code `awk -F '[,!]'` "," and "!" are fields seperators.

In this code "Hello" is a field seperator `awk -F Hello`

awk -v

The -v flag is used to specify the value of an AWK variable:

```
awk -v T = " Tut ori al: " '{print T,$0}' awk.txt
 >>> Tutorial: AWK is awesome
 >>> Tutorial: Let learn AWK together
 >>> Tutorial: AWK is used for text processing
```

awk Input

As we've already seen we can specify an input file for the awk command, but we can also specify more than one file at the same time :

```
awk '{print $0}' file1.txt file2.txt
```

We can also ommit the input file. In this case AWK will read from the input in the command line. We can use several inputs and then tape Ctrl+D to exit or (Ctrl+Z in Windows):

```
awk '{print $0}'
 one two three
 >>> two
 Yellow Blue Red
 >>> Blue
```

We can also specify the input from a file:

```
awk '{print $0} < file.txt
```

We can also take an input from a command with a vertical bar. Let try an example with the command `date`:

```
date
 >>> Thur Sept 12 11:12:05 CEST 2019
 date | awk '{print $2}'
 >>> Sept
```

awk Output

We can save the output in output.txt:

```
awk '{print $2}' awk.txt > out.txt
```

We can send the output to a program. Let use command `sort` to sort the output alphabetical

```
awk '{print NF,$0}' awk.txt |sort
 >>> AWK is awesome
 >>> AWK is used for text processi
 >>> Let learn AWK together
```



By **Nouha_Thabet**
cheatography.com/nouha-thabet/

Not published yet.
Last updated 12th December, 2019.

Sponsored by **Readable.com**
Measure your website readability!
https://readable.com