Table Of Contents

How To Use awk In Bash Scripting

Posted by Vivek Gite <vivek@nixcraft.com>

How do I use awk pattern scanning and processing language under bash scripts? Can you provide a few examples?

Awk is an excellent tool for building UNIX/Linux shell scripts. AWK is a programming language that is designed for processing text-based data, either in files or data streams, or using shell pipes. In other words you can combine awk with shell scripts or directly use at a shell prompt.

[1]

Print a Text File

```
awk '{ print }' /etc/passwd
```

OR

```
awk '{ print $0 }' /etc/passwd
```

Print Specific Field

Use : as the input field separator and print first field only i.e. usernames (will print the the first field. all other fields are ignored):

```
awk -F':' '{ print $1 }' /etc/passwd
```

Send output to sort command using a shell pipe:

```
awk -F':' '{ print $1 }' /etc/passwd | sort
```

Pattern Matching

You can only print line of the file if pattern matched. For e.g. display all lines from Apache log file if HTTP error code is 500 (9th field logs status error code for each http request):

```
awk '$9 == 500 { print $0}' /var/log/httpd/access.log
```

The part outside the curly braces is called the "pattern", and the part inside is the "action". The comparison operators include the ones from C:

```
== != < > <= >= ?:
```

If no pattern is given, then the action applies to all lines. If no action is given, then the entire line is printed. If "print" is used all by itself, the entire line is printed. Thus, the following are equivalent:

```
awk '$9 == 500 ' /var/log/httpd/access.log
awk '$9 == 500 {print} ' /var/log/httpd/access.log
awk '$9 == 500 {print $0} ' /var/log/httpd/access.log
```

Print Lines Containing tom, jerry AND vivek

Print pattern possibly on separate lines:

```
awk '/tom|jerry|vivek/' /etc/passwd
```

Print 1st Line From File

```
awk "NR==1{print;exit}" /etc/resolv.conf
awk "NR==$line{print;exit}" /etc/resolv.conf
```

Simply Arithmetic

You get the sum of all the numbers in a column:

```
awk '{total += $1} END {print total}' earnings.txt
```

Shell cannot calculate with floating point numbers, but awk can:

```
awk 'BEGIN {printf "%.3f\n", 2005.50 / 3}'
```

Call AWK From Shell Script

A shell script to list all IP addresses that accessing your website. This script use awk for processing log file and verification is done using shell script commands.

```
#!/bin/bash
d=$1
OUT=/tmp/spam.ip.$$
HTTPDLOG="/www/$d/var/log/httpd/access.log"
[ $# -eq 0 ] && { echo "Usage: $0 domain-name"; exit 999; }
if [ -f $HTTPDLOG ];
then
 awk '{print}' $HTTPDLOG >$OUT
 awk '{ print $1}' $OUT  |  sort -n | uniq -c | sort -n
else
 echo "$HTTPDLOG not found. Make sure domain exists and setup correctly."
fi
/bin/rm -f $OUT
```

AWK and Shell Functions

Here is another example. chrootCpSupportFiles() find out the shared libraries required by each program (such as perl / php-cgi) or shared library specified on the command line and copy them to destination. This code calls awk to print selected fields from the ldd output:

```

chrootCpSupportFiles() {
# Set CHROOT directory name
local BASE="$1"         # JAIL ROOT
local pFILE="$2"        # copy bin file libs

[ ! -d $BASE ] && mkdir -p $BASE || :

FILES="$(ldd $pFILE | awk '{ print $3 }' |egrep -v ^'\(')"
for i in $FILES
do
  dcc="$(dirname $i)"
  [ ! -d $BASE$dcc ] && mkdir -p $BASE$dcc || :
  /bin/cp $i $BASE$dcc
done

sldl="$(ldd $pFILE | grep 'ld-linux' | awk '{ print $1}')"
sldlsubdir="$(dirname $sldl)"
if [ ! -f $BASE$sldl ];
then
        /bin/cp $sldl $BASE$sldlsubdir
else
        :
fi
}
```

This function can be called as follows:

```
chrootCpSupportFiles /lighttpd-jail /usr/local/bin/php-cgi
```

AWK and Shell Pipes

List your top 10 favorite commands:

```
history | awk '{print $2}' | sort | uniq -c | sort -rn | head
```

Sample Output:

```
    172 ls
     144 cd
      69 vi
      62 grep
      41 dsu
      36 yum
      29 tail
      28 netstat
      21 mysql
      20 cat
```

whois cyberciti.com | awk '/Domain Expiration Date:/ { print $6"-"$5"-"$9 }'

Awk Program File

You can put all awk commands in a file and call the same from a shell script using the following syntax:

```
awk -f mypgoram.awk input.txt
```

Awk in Shell Scripts - Passing Shell Variables TO Awk

You can pass shell variables to awk using the -v option:

```

n1=5
n2=10
echo | awk -v x=$n1 -v y=$n2 -f program.awk
```

Assign the value n1 to the variable x, before execution of the program begins. Such variable values are available to the BEGIN block of an AWK program:

```
BEGIN{ans=x+y}
{print ans}
END{}
```