How To Use awk In Bash Scripting

by nixCraft on August 15, 2009 · 28 comments· LAST UPDATED August 14, 2009

in BASH Shell, CentOS, Debian / Ubuntu

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.

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{}
```


If you would like to be kept up to date with our posts, you can follow us on Twitter, Facebook, Google+, or even by subscribing to our RSS Feed.

Featured Articles:

- 30 Handy Bash Shell Aliases For Linux / Unix / Mac OS X
- Top 30 Nmap Command Examples For Sys/Network Admins
- 25 PHP Security Best Practices For Sys Admins
- 20 Linux System Monitoring Tools Every SysAdmin Should Know
- 20 Linux Server Hardening Security Tips
- Linux: 20 Iptables Examples For New SysAdmins
- Top 20 OpenSSH Server Best Security Practices
- Top 20 Nginx WebServer Best Security Practices
- 20 Examples: Make Sure Unix / Linux Configuration Files Are Free From Syntax Errors
- 15 Greatest Open Source Terminal Applications Of 2012 NEW
- My 10 UNIX Command Line Mistakes
- Top 10 Open Source Web-Based Project Management Software
- Top 5 Email Client For Linux, Mac OS X, and Windows Users
- The Novice Guide To Buying A Linux Laptop

{ 28 comments… read them below or add one }

1 alok August 16, 2009 at 12:27 pm

Thanks for this Helpful post, but is it possible to call bash user-defined function within awk.
eg:
!#/bin/bash
function Print_NATE (){
echo $1
}
awk '{ Print_name "ALOK' }'

I know the awk syntax is wrong, can you provide me small HOW TO on this.

Thanks

Reply

2 nixCraft August 16, 2009 at 1:07 pm

Use system()

```
echo|awk '{ system("date")}'
```

Reply

3 nfree August 16, 2009 at 2:22 pm

WTF?
```
41 dsu
```

Reply

4 nerdoug August 16, 2009 at 3:02 pm

some of my one liner awk tricks:

— To convert squid log timestamps to readable, sortable format:

```
gawk '{print strftime("%m/%d %H:%M:%S ",$1)" "substr($0,12,999)}' access.log > dated
```

— To avoid having to cut/paste the above, I have in my .profile file:

```
alias tim="gawk '{print strftime(\"%m/%d %H:%M:%S \",\$1)\" \"substr(\$0,12,999)}'"
```

— To use AWK to process comma separated data:

```
awk -F, '{print $1, "," $6}' excel-save.csv> extract.csv
```

— To count complex pattern occurrence

```
awk '{if (substr($2,1,4) == "2008" && $4 == "Exception") {print $1}}' test|grep -c ^
```

Reply

5 nixCraft August 16, 2009 at 3:09 pm

@nerdoug, Thanks for sharing your awk one-liners.

@ nfree, dsu is my own custom made tool to perform certain operations.

Reply

6 nerdoug August 16, 2009 at 3:21 pm

An alternate way to call awk programs (or “scripts”), in either Linux or cygwin:

have awk script start with the line:
#!/usr/bin/gawk -f

have bash calling sequences like:
./script input-file
./script -v param1=”08/15/” -v param2=”MISS/5″ dated > output-file

Reply

7 alok August 16, 2009 at 4:27 pm

Thanks Vivek Gite, but
the issue comes when we need to pass an argument to the user-define-functions.
Can you paste any example, which take any argument .

Thanks

Reply

8 nerdoug August 16, 2009 at 6:32 pm

Hope this illustrates passing arguments from bash to an awk script:

```
$ ./demo.sh
1 is a valid month number
4 is a valid month number
8 is a valid month number
12 is a valid month number
18 is not a valid month number
300 is not a valid month number
$ cat demo.sh
#!/bin/bash
# demonstrating how to pass a parameter from bash to an awk script
for tester in 1 4 8 12 18 300; do
./monthcheck.awk -v awkparam1=$tester monthlist
done
$ cat monthcheck.awk
#!/usr/bin/gawk -f
BEGIN {
   answer = "is not a valid month number"
}
{
   if ( $1 == awkparam1 ) {
      answer = "is a valid month number"
   }
}
END {
   print awkparam1  " " answer
}
delliott@delliott-lap2 /cygdrive/c/proj/jenson
$ cat monthlist
```

```
1
2
3
4
5
6
7
8
9
10
11
12
$ ./demo.sh
1 is a valid month number
4 is a valid month number
8 is a valid month number
12 is a valid month number
18 is not a valid month number
300 is not a valid month number
```

Reply

9 Ningappa August 18, 2009 at 5:22 am

Thanks vivek n nerdoug …. it looks awsome tut for newbies

Reply

10 Salih August 23, 2009 at 7:28 am

It is a very useful post.
Comments are also vey helpful.

Thanks

Reply

11 alok August 23, 2009 at 11:28 am

HI nerdoug
Thanks for the nice post ” Hope this illustrates passing arguments from bash to an awk script:”
But I am looking for just inverse case
“passing arguments form awk to bash user-define function.”

It will be a great help if you can come up with some example.

Thanks for your time and such a useful note.

–Alok

Reply

12 nerdoug August 23, 2009 at 5:57 pm

```
# ======================================================
# method 1A: returning a value from inline awk command via output stream
$ cat expenses
name date hotel breakfast lunch dinner beer
Doug 2009-07-20 159 10 15 32 44
Doug 2009-07-21 159 0 12 25 0
Doug 2009-07-22 159 8 15 41 87
```

```
Doug 2009-07-23 159 0 0 10 0
Doug 2009-07-24 0 11 15 0 0
$ cat checker-inline-awk.sh
#!/bin/bash
# bash script which calls awk to return a value
#    - here, value is biggest daily food expense in file
#    -also does arithmetic is awk, easier than in bash
#    -value is returned as the output of the awk command/script
#    -this sacrifices production of a regular output stream
#    -the first if skips over first data file line with column titles
#
bash_max=`awk '{if ($1 !="name") {t=$4+$5+$6; if (t>big) {big=t}}} END {print big}' expenses`
echo " value returned to bash is "$bash_max
if [ bash_max > 60 ] ; then
   echo " -daily food expense guideline violated"
else
   echo " -no food expense violation"
fi
$ ./checker-inline-awk.sh
 value returned to bash is 64
 -daily food expense guideline violated
#=====================================================
# method 1B: returning a value from awk script via output stream
$ cat checker-awk-script.sh
#!/bin/bash
# bash script which calls awk script to return a value
bash_max=`./scan.awk expenses`
echo " value returned to bash is "$bash_max
if [ bash_max > 60 ] ; then
   echo " -daily food expense guideline violated"
else
   echo " -no food expense violation"
fi
$ cat scan.awk
#!/usr/bin/awk -f
# awk script to return largest food expenditure from input file
{
  if ($1 !="name") {
     t=$4+$5+$6
     if (t>big) {
        big=t
     }
  }
}
END {print big}
$ ./checker-awk-script.sh
 value returned to bash is 64
 -daily food expense guideline violated
#=================================================
# method 2: returning value via temporary disk file, with normal output stream
$ cat checker-disk.sh
#!/bin/bash
# bash script which calls awk to return a value
#    -value is returned via a temporary disk file
#    -still allows production of a regular output stream
# run the awk scan creating summary file, and max value in disk file
./scan-disk.awk expenses > summary
# retrieve value written to disk
bash_max=`cat big.txt`
if [ bash_max > 60 ] ; then
   echo "*** daily food expense guideline violated"
   echo "    see following details"
   cat summary
else
   echo " -no food expense violation"
fi
$ cat scan-disk.awk
```

```
#!/usr/bin/awk -f
# awk script to return largest food expenditure from input file
# -by writing resulting value to a temporary disk file
# -also produce normal output stream
{
  if ($1 !="name") {
     t=$4+$5+$6
     if (t>big) {
        big=t
     }
     # produce reformatted output stream
     print $2 " food=" t " beer= "$7
  }
}
END {
   print big > "big.txt"
}
delliott@delliott-lap2 /cygdrive/c/proj
$ ./checker-disk.sh
*** daily food expense guideline violated
    see following details
2009-07-20 food=57 beer= 44
2009-07-21 food=37 beer= 0
2009-07-22 food=64 beer= 87
2009-07-23 food=10 beer= 0
2009-07-24 food=26 beer= 0
#=========================================================
# Method I can't get to work
#
# I was hoping that using this command to embed a shell command in awk would work...
#       t = system("export bash_max=123")
#
# but I can't get it to work, perhaps due to too many levels of child processes?
#
```

Reply

13 nixCraft August 23, 2009 at 6:07 pm

> but I can’t get it to work, perhaps due to too many levels of child processes?

Yes, you need to keep everything in same shell. Awk calls sh whenever system() is used. To OP, you better use perl or python if you need really complicated stuff.

Reply

14 arka prava chandra April 23, 2010 at 8:31 pm

dear sir,

I would like to know how could I use awk with conditional statement.

for ex . -

if there is two condition (let 14 , 17 (which are in 3rd column)) & another 2 conditions(let ab,cd (which are in 4th & 6th column respectively)).
& it will porceed if condition ”14de” or ”17de” are true .

(all the data’s are in ”xyz.txt” file)
I have made like :-

‘cat xyz.txt | awk -F”,” ‘{ if(((substr($3,1,2)==14) || (substr($3,1,2)==17))) && (substr($4,1,4)==”\”ab\”") &&

(substr($6,1,4)!="\"cd\"")) print($4) }' "| wc -l'

.

Reply

15 nerdoug April 25, 2010 at 12:05 am

I have a comment on your bash, and a couple on your awk.

bash: doing cat infile | awk '{stuff}' > outfile causes each line in the input file to be processed twice, once by cat, and once by awk.
if instead you do awk '{stuff}' infile > outfile there's only one pass through the file. If you're processing big files, this can be significant.

awk: I'm not sure if your boolean logic will work, because I don't understand whatt you meant by the goal of proceeding if condition "14de" or "17de" are true. You can do some of the filtering in a regular expression, and perhaps all of it depending on the formating of your data. for example, I think you can replace the first part of your if logic like this:
awk -F, /^1[47]/ '{if((substr($4,1,4)=="\"ab\"") && (substr($6,1,4)!="\"cd\"")) {print $4 } }' xyz.txt | wc -l

I don't think the last single quote and the last double quote are needed in your awk.

hope this helps.

Reply

16 Philippe Petrinko December 16, 2010 at 6:03 pm

@Vivek.
Thanks for your topic.

BTW, reagrding passing shell variables to [awk],
that can be used in BEGIN block,
one can also use ENVIRON array:

Try:

export myawkvar=something; awk 'BEGIN {print "BEGIN:" ENVIRON["myawkvar"] ":"}'

Would you add this builtin function to your topic?

–P

Reply

17 Vicky June 26, 2011 at 8:47 am

Thank u Vivek ji.

Reply

18 Dinesh July 29, 2011 at 10:02 am

a directory contains few folders where my files are located(all files in all folders need to be processed).now,how to use awk recursively for all files. in grep we have

$grep -r string *.asc
wat abt awk..?

Reply

19 Anantha September 10, 2012 at 8:09 am

Put it in a for loop

```
for i in *
do
 awk '{ your awk commands } $i
done
```

Reply

20 Philippe Petrinko September 10, 2012 at 9:32 am

No, Anantha, this won't do.
Again, think of Unix philosophy: Each command should do only one thing, and do it well.
http://en.wikipedia.org/wiki/Unix_philosophy

Most of the time, this implies using pipes.

Precisely in this case, each time a program needs to traverse directories recursively, the right tool is [find]. Then pipe its result into [xargs], that will call [awk].

Search The Fantastic Web with these commands names (unix find xargs awk) and it's up to you now => http://lmgtfy.com/?q=unix+find+xargs+awk

–P

Reply

21 ITtuition.com August 17, 2011 at 1:05 pm

Great Post and comment! Thanks for sharing your awknowledge!

Reply

22 Stanislav August 31, 2011 at 1:43 pm

Hi to all!
I have a question on using awk in bash scripts.
I am completely confused.

I have a file
voc.txt:
book
help

and a script generate1.sh:
voc=$1
awk '{print $1}' $voc

When I run
>./generate1.sh voc.txt

I get
book
help
as expected.

But when I run the script
generate.sh:
voc=$1
awk '{print " word" NR " = (any * '" $1 "' space @increment" NR ") * ;"}' $voc
I get
word1 = (any * voc1.txt space @increment1) * ;
word2 = (any * voc1.txt space @increment2) * ;

If I put a backslash before $, I get

word1 = (any * '$1′ space @increment1) * ;
word2 = (any * '$1′ space @increment2) * ;

How can I get

word1 = (any * 'book' space @increment1) * ;
word2 = (any * 'help' space @increment2) * ;

?

Reply

23 Brad G. September 9, 2011 at 11:46 am

Very nice site with good useful info. Thank you.

Reply

24 walnutpony123 February 3, 2012 at 7:53 pm

Hi Can some one help me put together bash script that runs coulmns
1 2 3 4 5 6
and not 1
2
3
4
5
6

with column headers column1 = tj/art
column2=tj/art
column3=tj
column4=art
column5=tj
column6=art

and then redirect to a file >>

Reply

25 mini_csreddy March 6, 2012 at 11:43 am

Hi All,

I have a shell script program which works in Unix platform and able to load data in staging and interface tables, but when it migrated to other instance of linux platform, its not working and not able to load data.

Is there any change to do with awk and sed commands in Linux ?

Please help in this regard…

Please find is shell script program

```
#print "Dollar 1 $1"
org_id=$(print $1 | awk '{print $9}' | sed 's/"//g')
print $org_id
LOGIN=$(print $1 | awk '{print $3}' | sed 's/"//g' | sed 's/FCP_LOGIN=//')
datafile=FORECAST'.'csv
print $datafile
print $GT_TOP
forecast_file=$GT_TOP/bin/$org_id$datafile
print $forecast_file
#  sed 's/
//' $forecast_file  > GT_TOP/bin/$org_id.dat
#  mv  $GT_TOP/bin/$org_id.dat $forecast_file
dos2ux $forecast_file> GT_TOP/bin/test.csv
rm $forecast_file
mv $GT_TOP/bin/test.csv $forecast_file
chmod 777 $forecast_file
sqlldr $LOGIN control=$GT_TOP/bin/GB_FORECAST_UPLOAD.ctl data=$forecast_file
sqlplus -s $LOGIN << EOF
BEGIN
gb_forecast_prog_pkg.gb_forecast_pr;
commit;
END;
/
Exit;
EOF
```

Reply

26 madhuri October 15, 2013 at 9:16 am

I have a file with the content as
/a/b/sometext1
/a/b/c/sometex2
/a/b/d

I want to separate out /a/b fields from all the lines.
how can I use awk command.

Reply

27 Dinesh October 15, 2013 at 1:07 pm

$ cat file
/a/b/sometext1
/a/b/c/sometex2
/a/b/d

```
$ awk -F/ '{print "/"$2"/"$3}' file
/a/b
/a/b
/a/b

$ awk -F/ '{print $2,$3}' file
a b
a b
a b
```

Reply

28 Dinesh October 15, 2013 at 1:07 pm

```
$ cat file
/a/b/sometext1
/a/b/c/sometex2
/a/b/d

$ awk -F/ '{print "/"$2"/"$3}' file
/a/b
/a/b
/a/b

$ awk -F/ '{print $2,$3}' file
a b
a b
a b
```

Reply

Leave a Comment

Name *

E-mail *

Website

You can use these HTML tags and attributes for your code and commands: <u> <kbd> <blockquote> <pre> <a href="" title="">

Notify me of followup comments via e-mail

Submit

Tagged as: apache log, arithmetic, awk, awk command, awk print, bash scripts, bash shell awk, curly braces, data streams, field logs, field separator, floating point numbers, input field, korn shell awk, pipes, shell script, shell scripts, sort command, status error

Previous Faq: Adobe Photoshop Alternatives For Linux Desktop

Next Faq: Samba: Allow Domain Controllers Create Machine Trust Accounts On-the-Fly

- **Related Faqs**
  -  HowTo: Read a File Line By Line Using awk
  -  Bash Script Replace Empty Spaces String

- Awk Print Line After A Matching /regex/

- Shell: How To Remove Duplicate Text Lines

- Howto: Call Shell Script function In echo statement

- UNIX / Linux: awk Add Two Numbers

- Diagnosing a dos Attack Under Linux

- Warning: bad syntax, perhaps a bogus ‘-’? See /usr/share/doc/procps-3.2.7/FAQ

- Explains: echo Command (echo $"string") Double-quoted String Preceded By a Dollar Sign

- Linux / Unix AWK: Read a Text File

- **Latest posts from our blog**
  - Valve SteamOS: A Linux-based Gaming Operating System Announced
  - Download of the day: Half-Life 2 For Steam on Linux
  - Download of The Day: Debian Linux 7 (Wheezy)
  - Apache / Nginx: Visualize Web Server Access Log In Real Time
  - Amazon AWS Route 53 GEO DNS Configurations