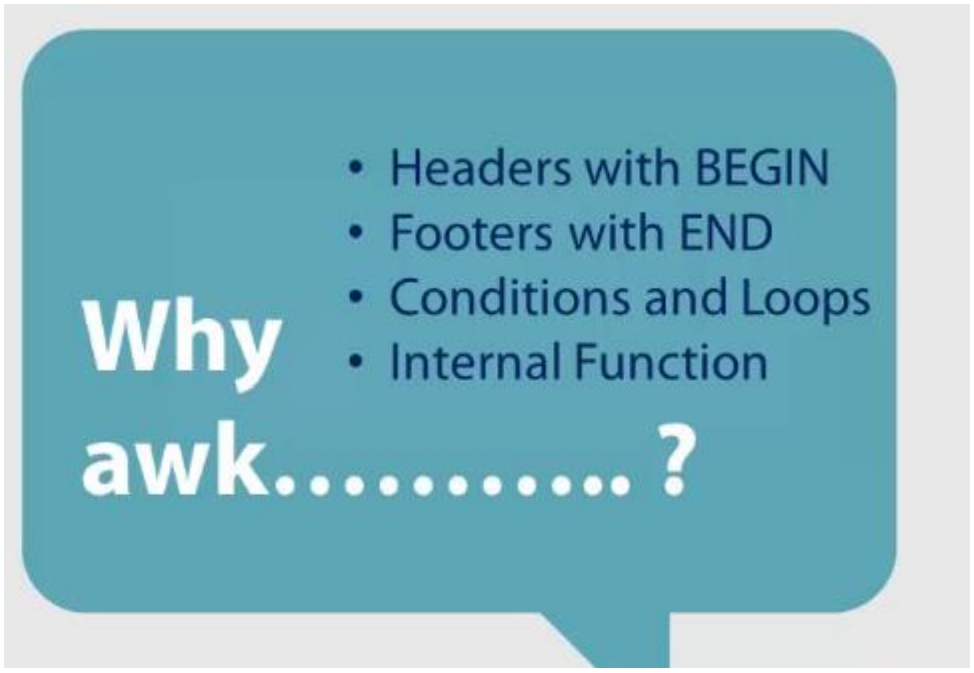
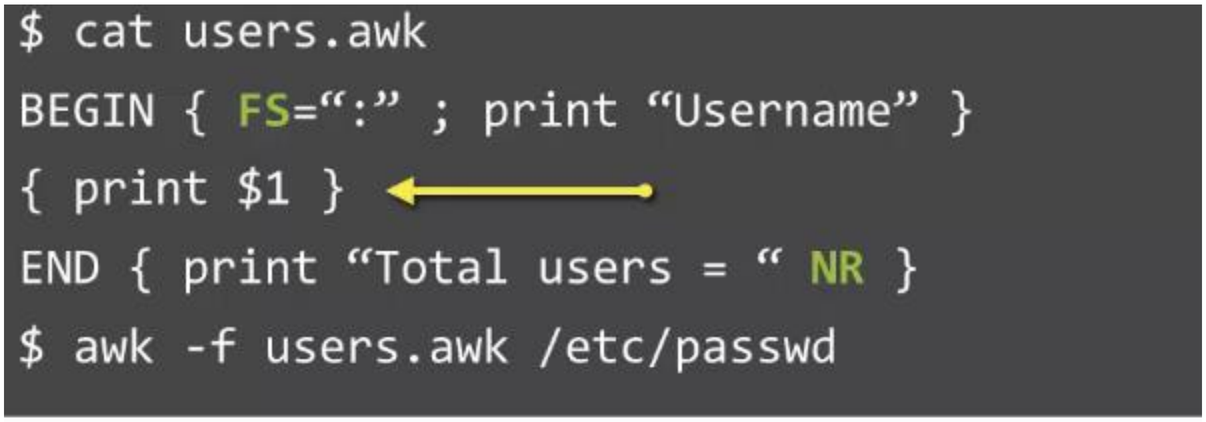
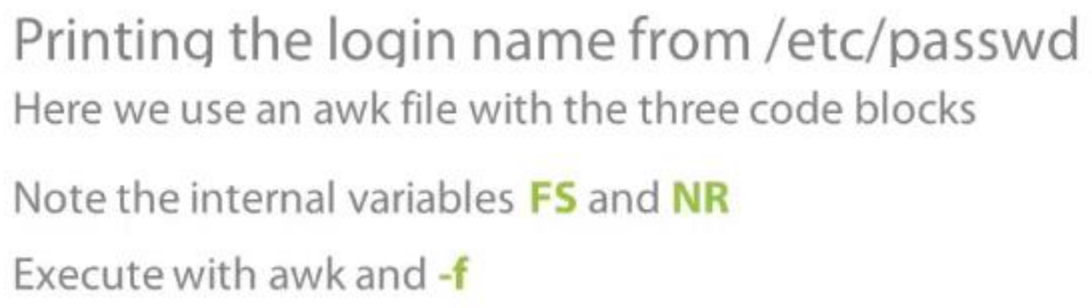

Why awk.........?

- Headers with BEGIN
- Footers with END
- Conditions and Loops
- Internal Function

FS field seperator



```
$ cat users.awk
BEGIN { FS=“:” ; print “Username” }
{ print $1 }
END { print “Total users = “ NR }
$ awk -f users.awk /etc/passwd
```

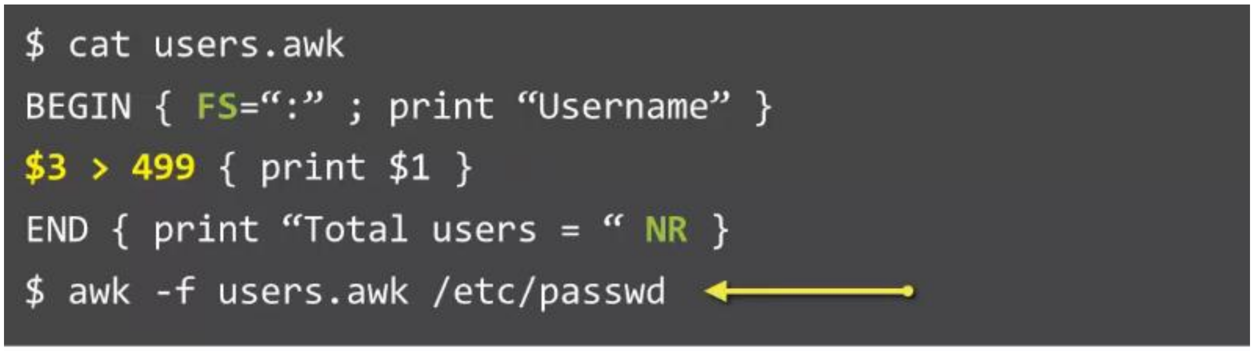


Printing the login name from /etc/passwd

Here we use an awk file with the three code blocks

Note the internal variables **FS** and **NR**

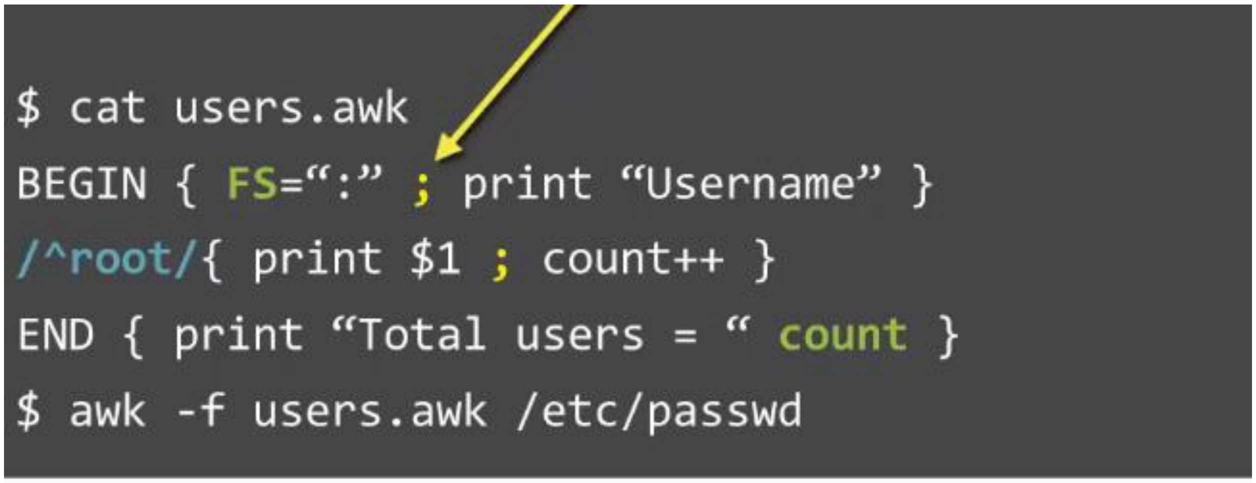
Execute with awk and **-f**



```
$ cat users.awk
BEGIN { FS=":" ; print "Username" }
$3 > 499 { print $1 }
END { print "Total users = " NR }
$ awk -f users.awk /etc/passwd
```

Print where the UID is greater than 499

Adding the range in front of the main block we can use field values to control the display of row



```
$ cat users.awk
BEGIN { FS=":" ; print "Username" }
/^root/{ print $1 ; count++ }
END { print "Total users = " count }
$ awk -f users.awk /etc/passwd
```

Counting returned rows

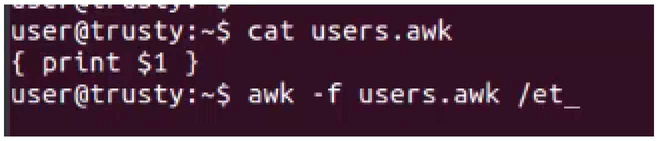
However the **NR** variable still shows all lines in the password file

We add the **count** variable to show processed lines

Semi-colon is used to separate commands

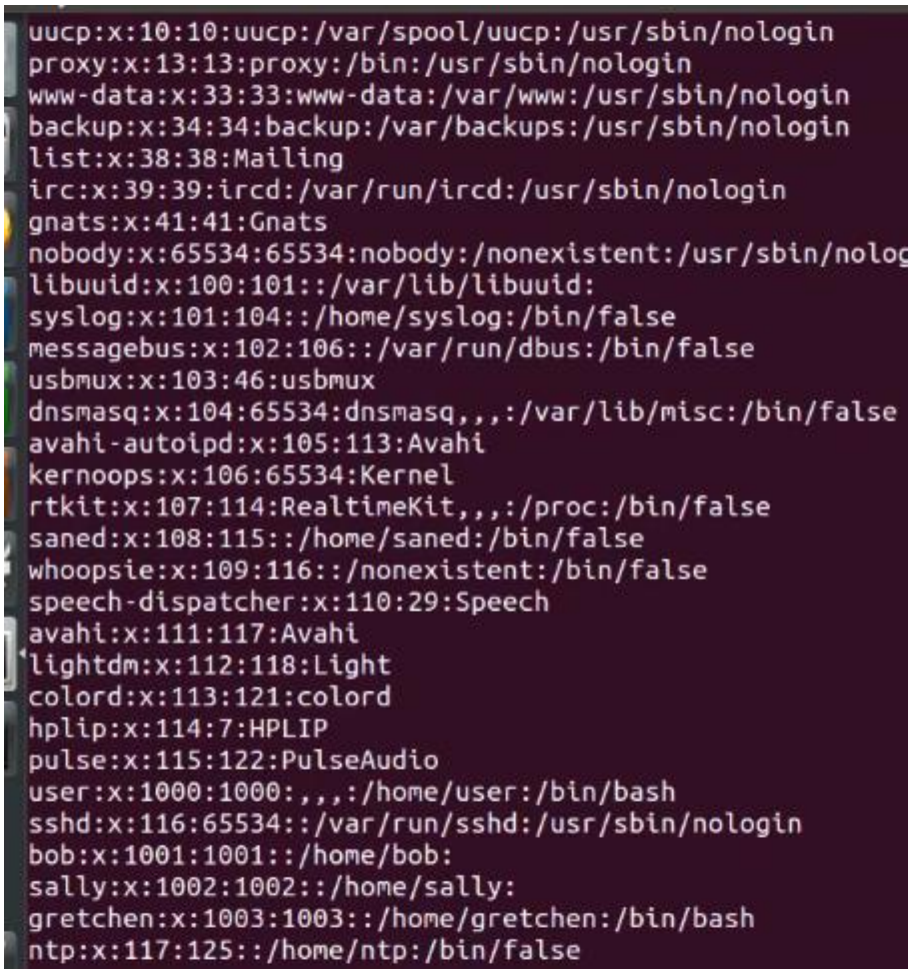
Basics Of Awk

Here we are just displaying the lines of file "etc/password"



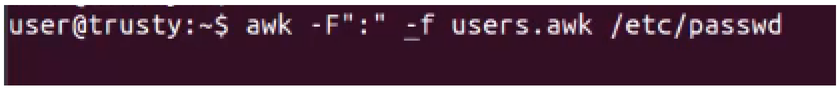
```
user@trusty:~$ cat users.awk
{ print $1 }
user@trusty:~$ awk -f users.awk /et_
```

Output



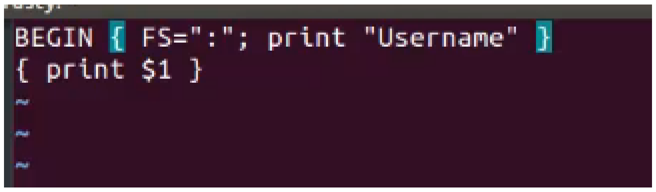
```
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing
irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nolog
libuuid:x:100:101::/var/lib/libuuid:
syslog:x:101:104::/home/syslog:/bin/false
messagebus:x:102:106::/var/run/dbus:/bin/false
usbmux:x:103:46:usbmux
dnsmasq:x:104:65534:dnsmasq,,,:/var/lib/misc:/bin/false
avahi-autoipd:x:105:113:Avahi
kernoops:x:106:65534:Kernel
rtkit:x:107:114:RealtimeKit,,,:/proc:/bin/false
saned:x:108:115::/home/saned:/bin/false
whoopsie:x:109:116::/nonexistent:/bin/false
speech-dispatcher:x:110:29:Speech
avahi:x:111:117:Avahi
lightdm:x:112:118:Light
colord:x:113:121:colord
hplip:x:114:7:HPLIP
pulse:x:115:122:PulseAudio
user:x:1000:1000:,,,:/home/user:/bin/bash
sshd:x:116:65534::/var/run/sshd:/usr/sbin/nologin
bob:x:1001:1001::/home/bob:
sally:x:1002:1002::/home/sally:
gretchen:x:1003:1003::/home/gretchen:/bin/bash
ntp:x:117:125::/home/ntp:/bin/false
```

Instead we can also specify field separator from command line but as we don't specify earlier the entire line will be displayed using –F

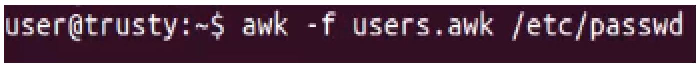


```
user@trusty:~$ awk -F":" -f users.awk /etc/passwd
```

INSTEAD WE CAN MENTION IT IN THE users.aawk file also in the header section

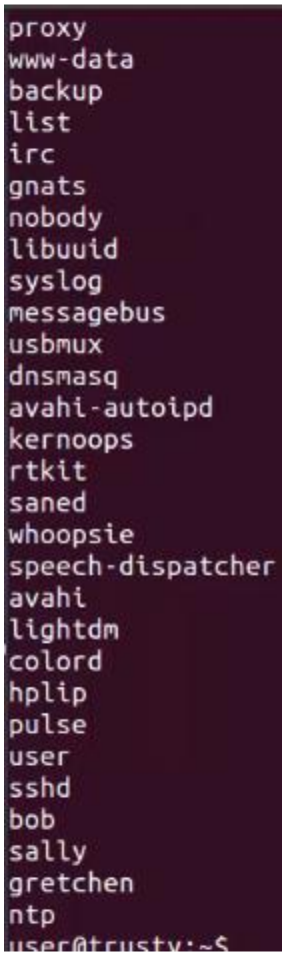


```
BEGIN { FS=":"; print "Username" }
{ print $1 }
~
~
~
```

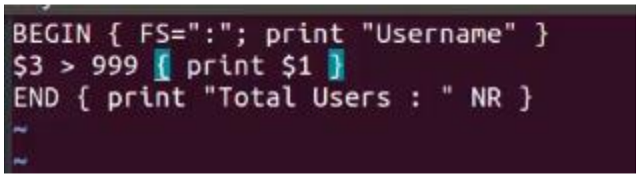


```
user@trusty:~$ awk -f users.awk /etc/passwd
```

OUTPUT

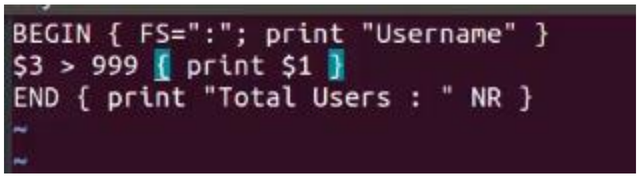


```
proxy
www-data
backup
list
irc
gnats
nobody
libuuid
syslog
messagebus
usbmux
dnsmasq
avahi-autoipd
kernoops
rtkit
saned
whoopsie
speech-dispatcher
avahi
lightdm
colord
hplip
pulse
user
sshd
bob
sally
gretchen
ntp
user@trusty:~$
```



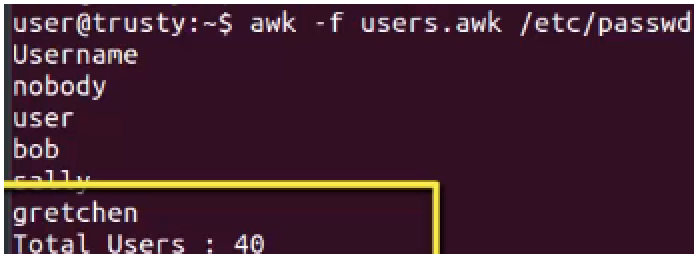
```
BEGIN { FS=":"; print "Username" }
$3 > 999 { print $1 }
END { print "Total Users : " NR }
~
~
```

In the above code we are filtering the 4 column which is username greater than 999 and printing total users as footer

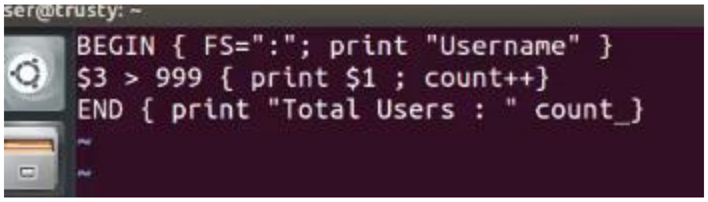


```
BEGIN { FS=":"; print "Username" }
$3 > 999 { print $1 }
END { print "Total Users : " NR }
~
~
```

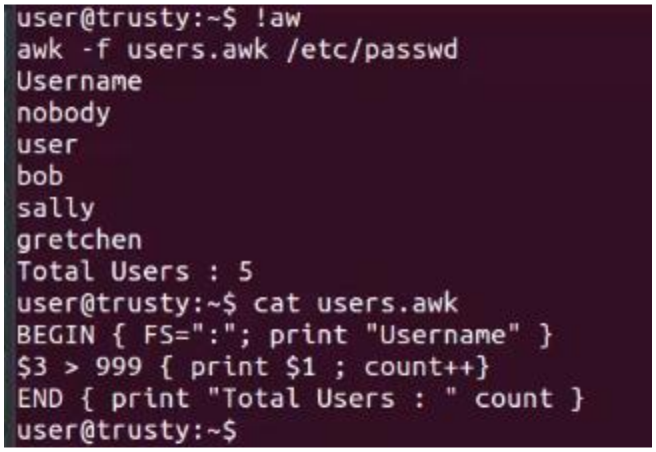
Output



```
user@trusty:~$ awk -f users.awk /etc/passwd
Username
nobody
user
bob
sally
gretchen
Total Users : 40
```

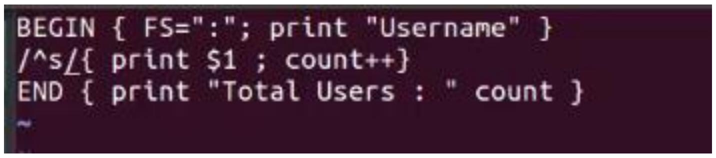


```
BEGIN { FS=":"; print "Username" }
$3 > 999 { print $1 ; count++}
END { print "Total Users : " count_}
~
~
```

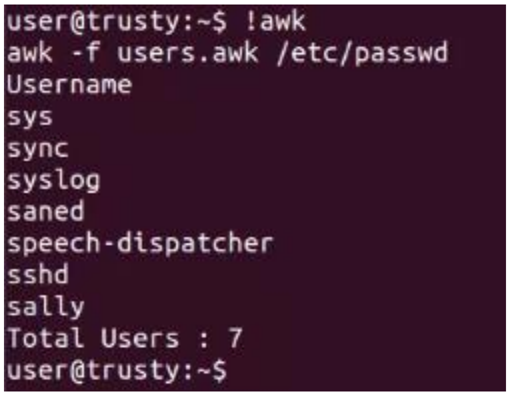


```
user@trusty:~$ !aw
awk -f users.awk /etc/passwd
Username
nobody
user
bob
sally
gretchen
Total Users : 5
user@trusty:~$ cat users.awk
BEGIN { FS=":"; print "Username" }
$3 > 999 { print $1 ; count++}
END { print "Total Users : " count }
user@trusty:~$
```

We can also use lines begin with as in grep

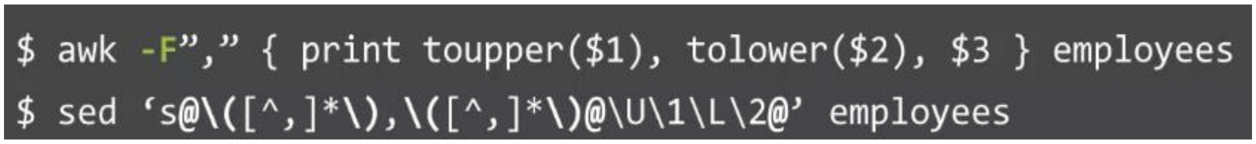


```
BEGIN { FS=":"; print "Username" }
/^s/{ print $1 ; count++}
END { print "Total Users : " count }
~
```



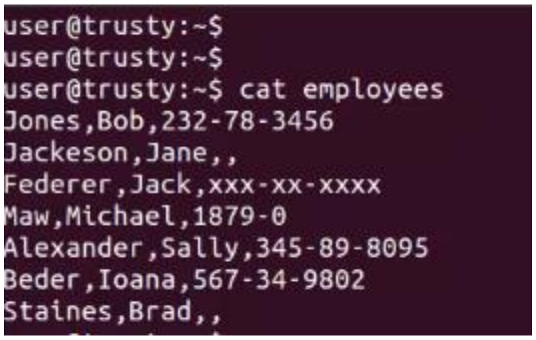
```
user@trusty:~$ !awk
awk -f users.awk /etc/passwd
Username
sys
sync
syslog
saned
speech-dispatcher
sshd
sally
Total Users : 7
user@trusty:~$
```

We can see awk is simple to replace than sed

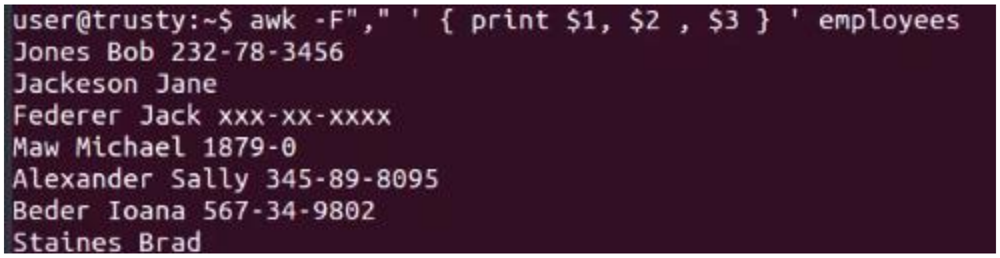


```
$ awk -F”,” { print toupper($1), tolower($2), $3 } employees
$ sed ‘s@\([^,]*\),\([^,]*\)@\U\1\L\2@’ employees
```

Input

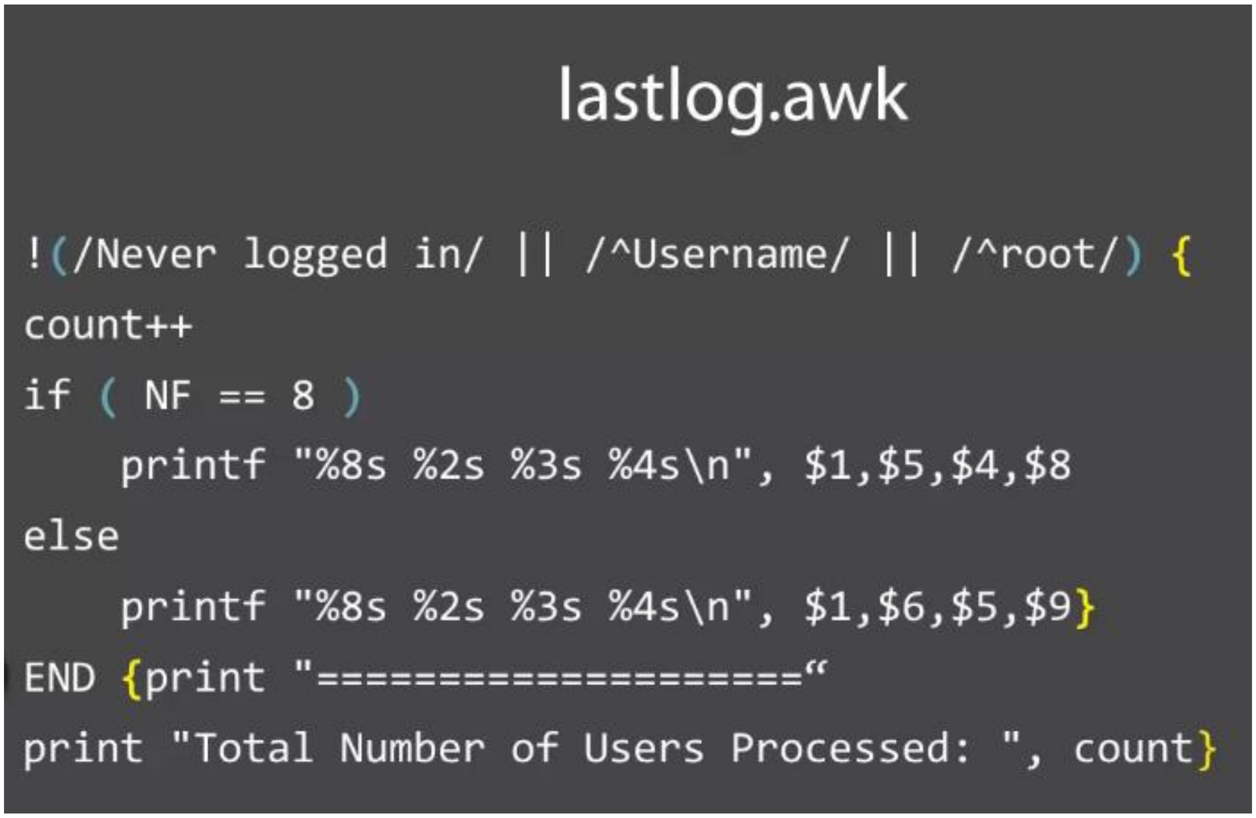


```
user@trusty:~$
user@trusty:~$
user@trusty:~$ cat employees
Jones,Bob,232-78-3456
Jackeson,Jane,,
Federer,Jack,xxx-xx-xxxx
Maw,Michael,1879-0
Alexander,Sally,345-89-8095
Beder,Ioana,567-34-9802
Staines,Brad,,
```



```
user@trusty:~$ awk -F"," ' { print $1, $2 , $3 } ' employees
Jones Bob 232-78-3456
Jackeson Jane
Federer Jack xxx-xx-xxxx
Maw Michael 1879-0
Alexander Sally 345-89-8095
Beder Ioana 567-34-9802
Staines Brad
```

Find user login details



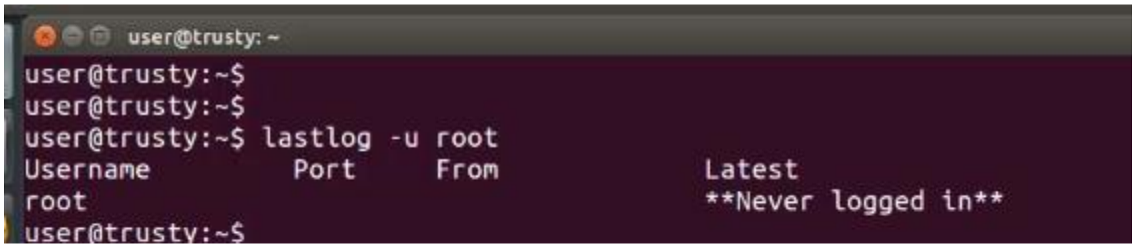
lastlog.awk

```
!(/Never logged in/ || /^Username/ || /^root/) {
count++
if ( NF == 8 )
    printf "%8s %2s %3s %4s\n", $1,$5,$4,$8
else
    printf "%8s %2s %3s %4s\n", $1,$6,$5,$9}
END {print "====================“
print "Total Number of Users Processed: ", count}
```

NF number of fields(column)

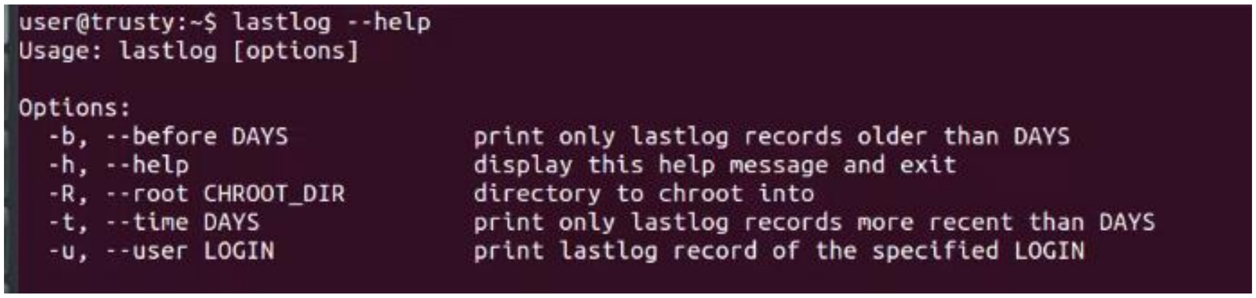
8s means 8 character string

Use lastlog command to check for last login for various users like root user etc

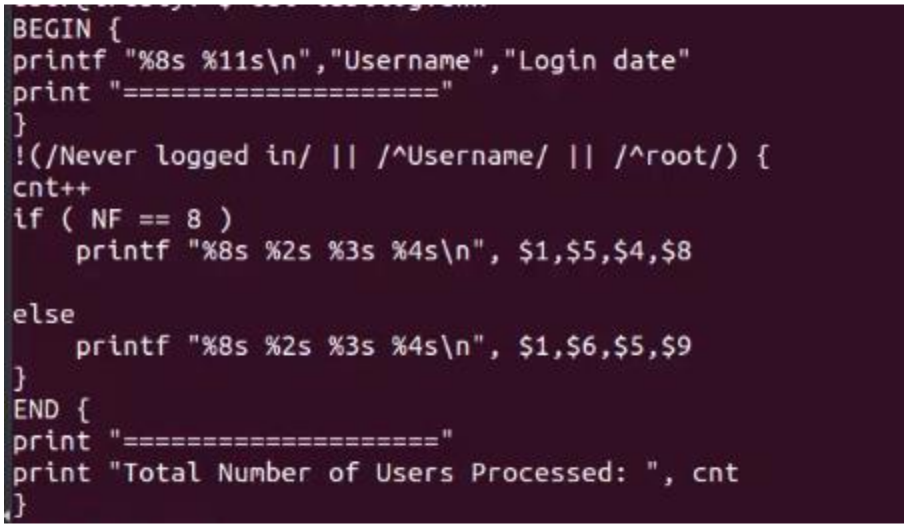


```
user@trusty:~$
user@trusty:~$
user@trusty:~$ lastlog -u root
Username         Port     From             Latest
root                                       **Never logged in**
user@trusty:~$
```

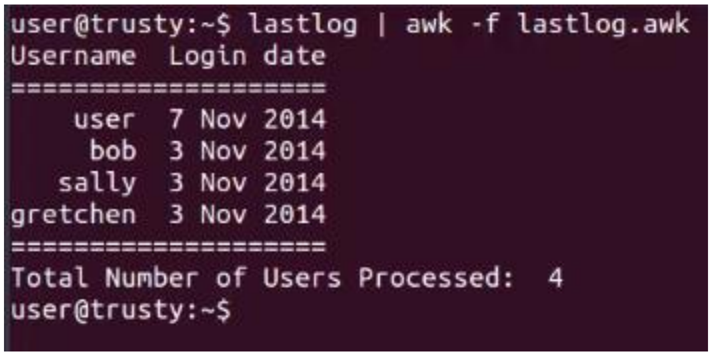
Use help command to see various options



```
user@trusty:~$ lastlog --help
Usage: lastlog [options]

Options:
  -b, --before DAYS             print only lastlog records older than DAYS
  -h, --help                    display this help message and exit
  -R, --root CHROOT_DIR         directory to chroot into
  -t, --time DAYS               print only lastlog records more recent than DAYS
  -u, --user LOGIN              print lastlog record of the specified LOGIN
```



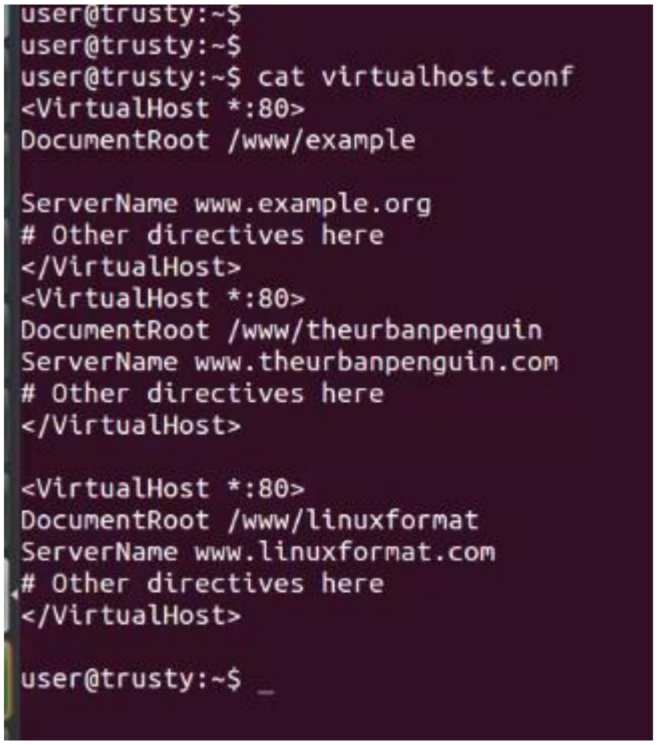
```
BEGIN {
printf "%8s %11s\n","Username","Login date"
print "===================="
}
!(/Never logged in/ || /^Username/ || /^root/) {
cnt++
if ( NF == 8 )
    printf "%8s %2s %3s %4s\n", $1,$5,$4,$8

else
    printf "%8s %2s %3s %4s\n", $1,$6,$5,$9
}
END {
print "===================="
print "Total Number of Users Processed: ", cnt
}
```



```
user@trusty:~$ lastlog | awk -f lastlog.awk
Username  Login date
====================
    user  7 Nov 2014
     bob  3 Nov 2014
   sally  3 Nov 2014
gretchen  3 Nov 2014
====================
Total Number of Users Processed:  4
user@trusty:~$
```

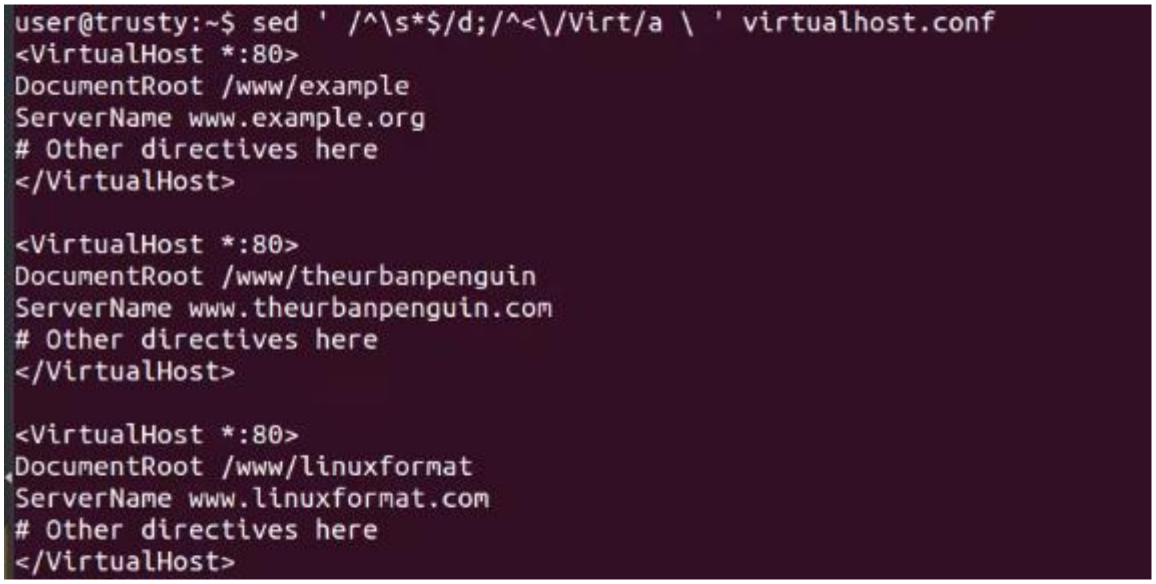
DISPLAYING RECORDS FROM XML FILE USING AWK

STEP 1 CLEANING INPUT DATA



```
user@trusty:~$
user@trusty:~$
user@trusty:~$ cat virtualhost.conf
<VirtualHost *:80>
DocumentRoot /www/example

ServerName www.example.org
# Other directives here
</VirtualHost>
<VirtualHost *:80>
DocumentRoot /www/theurbanpenguin
ServerName www.theurbanpenguin.com
# Other directives here
</VirtualHost>

<VirtualHost *:80>
DocumentRoot /www/linuxformat
ServerName www.linuxformat.com
# Other directives here
</VirtualHost>

user@trusty:~$ _
```

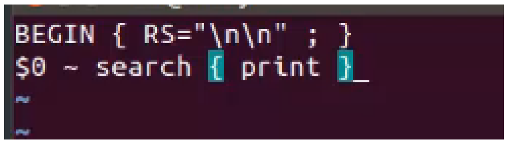
STEP2: CLEANING VIRTUAL DATA USING SED

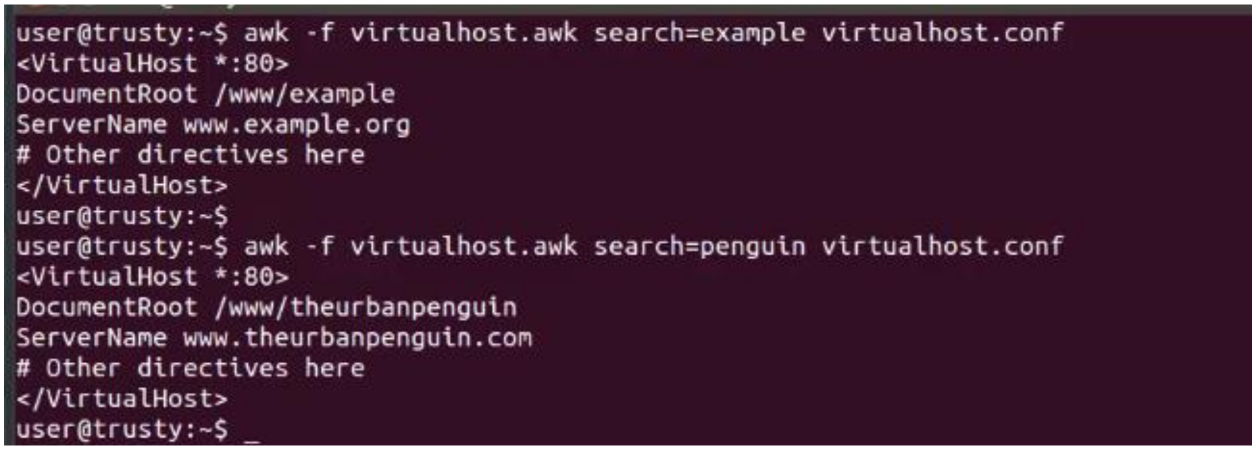


```
user@trusty:~$ sed ' /^\s*$/d;/^<\/Virt/a \ ' virtualhost.conf
<VirtualHost *:80>
DocumentRoot /www/example
ServerName www.example.org
# Other directives here
</VirtualHost>

<VirtualHost *:80>
DocumentRoot /www/theurbanpenguin
ServerName www.theurbanpenguin.com
# Other directives here
</VirtualHost>

<VirtualHost *:80>
DocumentRoot /www/linuxformat
ServerName www.linuxformat.com
# Other directives here
</VirtualHost>
```

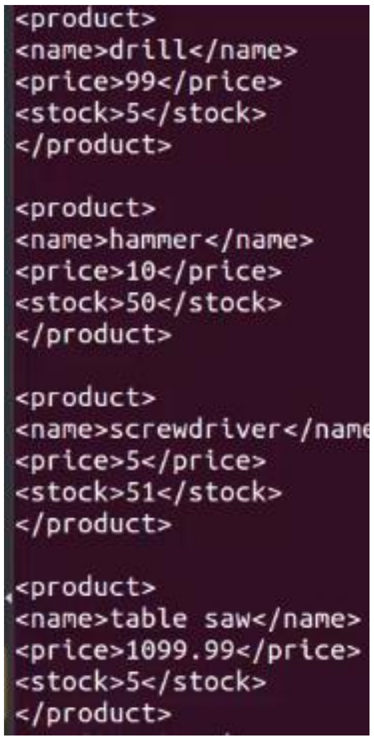
STEP3 :AWK FIEL

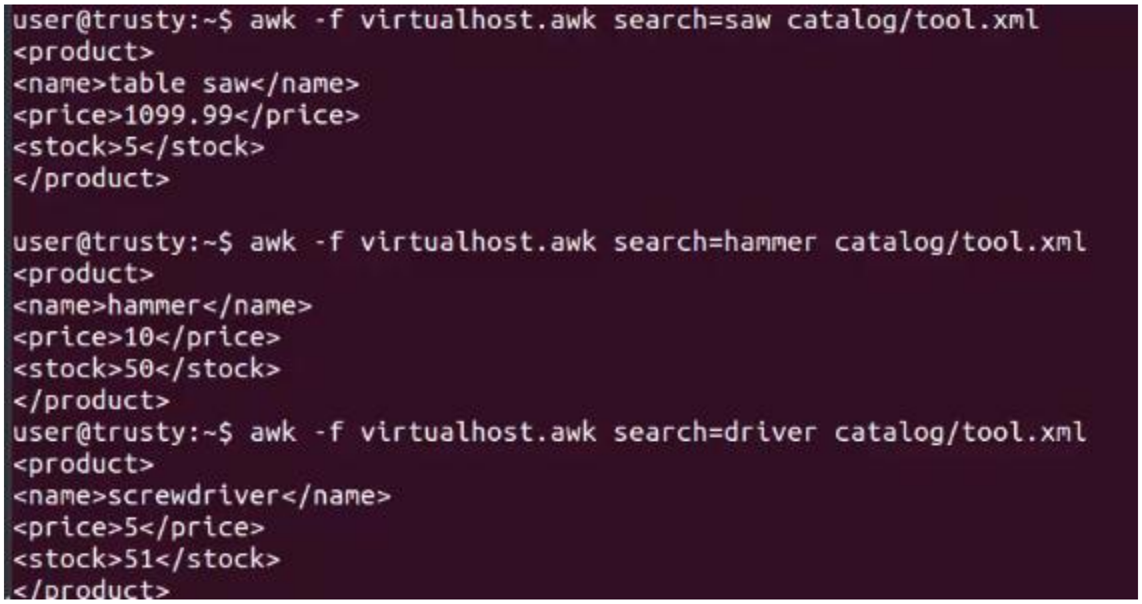
WE ARE GIVING RECORD SEPERATOR AS 2 BLANK LINE AND FIND THE LINE WHICH MATCHES THE SEARCH FOR WHICH VALUE WILL BE GIVEN ON FLY


```
BEGIN { RS="\n\n" ; }
$0 ~ search { print }_
~
~
```

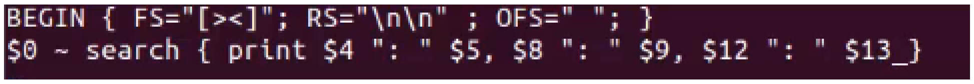

```
user@trusty:~$ awk -f virtualhost.awk search=example virtualhost.conf
<VirtualHost *:80>
DocumentRoot /www/example
ServerName www.example.org
# Other directives here
</VirtualHost>
user@trusty:~$
user@trusty:~$ awk -f virtualhost.awk search=penguin virtualhost.conf
<VirtualHost *:80>
DocumentRoot /www/theurbanpenguin
ServerName www.theurbanpenguin.com
# Other directives here
</VirtualHost>
user@trusty:~$ _
```

INPUT FILE

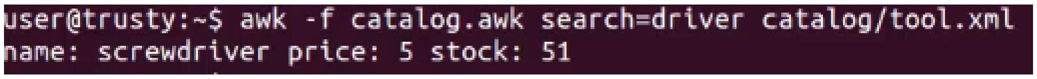

```
<product>
<name>drill</name>
<price>99</price>
<stock>5</stock>
</product>

<product>
<name>hammer</name>
<price>10</price>
<stock>50</stock>
</product>

<product>
<name>screwdriver</name
<price>5</price>
<stock>51</stock>
</product>

<product>
<name>table saw</name>
<price>1099.99</price>
<stock>5</stock>
</product>
```



```
user@trusty:~$ awk -f virtualhost.awk search=saw catalog/tool.xml
<product>
<name>table saw</name>
<price>1099.99</price>
<stock>5</stock>
</product>

user@trusty:~$ awk -f virtualhost.awk search=hammer catalog/tool.xml
<product>
<name>hammer</name>
<price>10</price>
<stock>50</stock>
</product>
user@trusty:~$ awk -f virtualhost.awk search=driver catalog/tool.xml
<product>
<name>screwdriver</name>
<price>5</price>
<stock>51</stock>
</product>
```



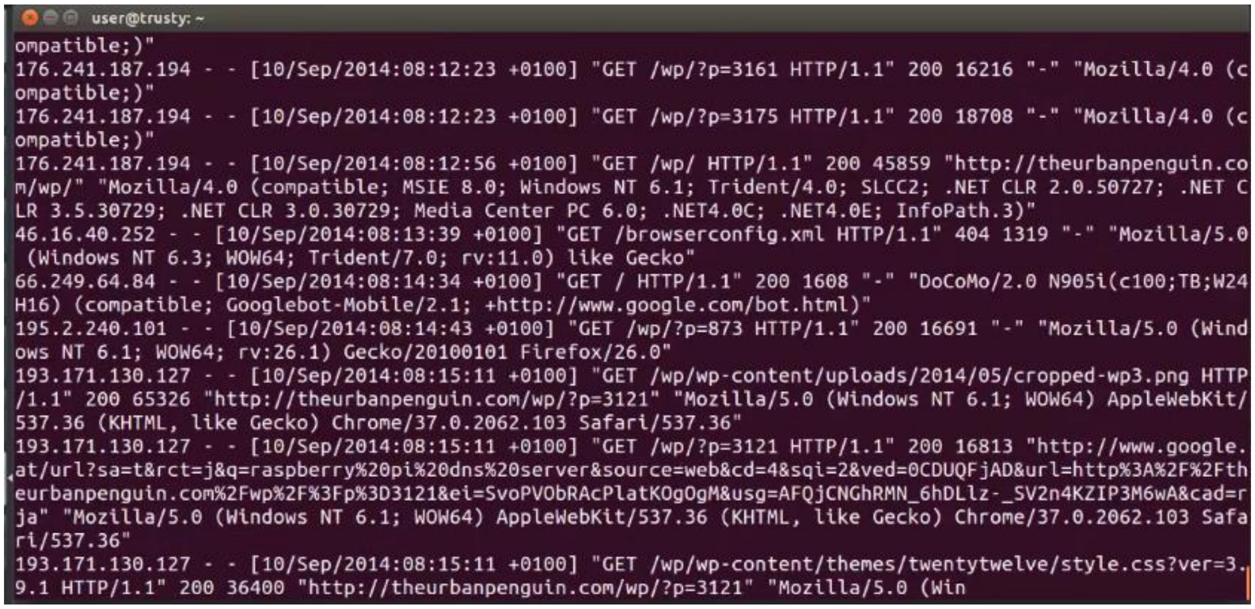
```
BEGIN { FS="[><]"; RS="\n\n" ; OFS=" "; }
$0 ~ search { print $4 ": " $5, $8 ": " $9, $12 ": " $13_}
```



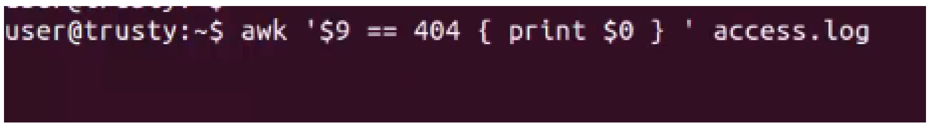
```
user@trusty:~$ awk -f catalog.awk search=driver catalog/tool.xml
name: screwdriver price: 5 stock: 51
```

ANALYSE WEB LOG USING AWK

SAMPLE DATA

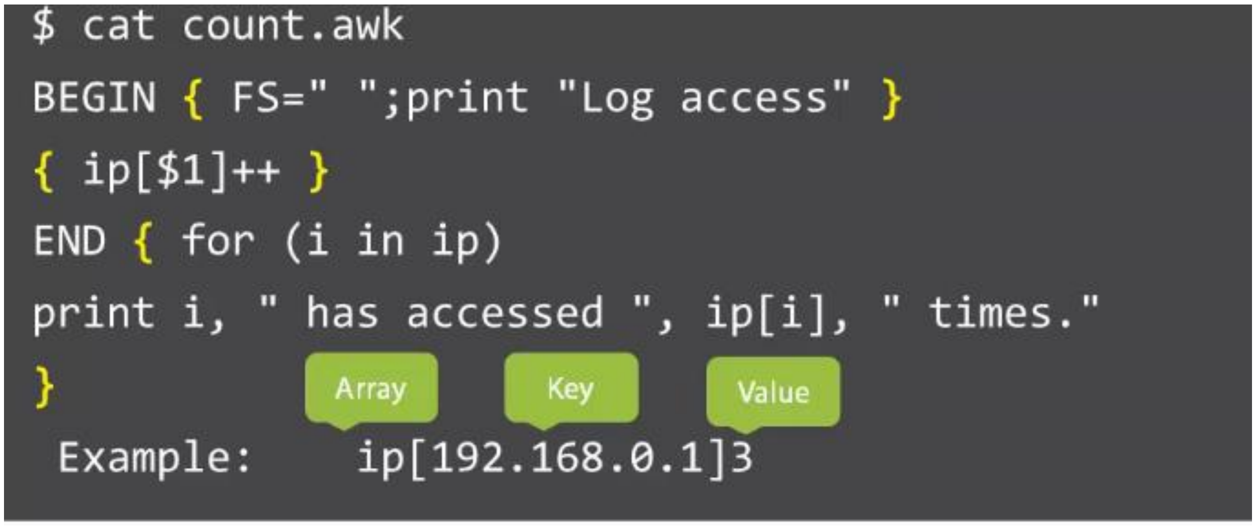


```
user@trusty: ~
ompatible;)"
176.241.187.194 - - [10/Sep/2014:08:12:23 +0100] "GET /wp/?p=3161 HTTP/1.1" 200 16216 "-" "Mozilla/4.0 (c
ompatible;)"
176.241.187.194 - - [10/Sep/2014:08:12:23 +0100] "GET /wp/?p=3175 HTTP/1.1" 200 18708 "-" "Mozilla/4.0 (c
ompatible;)"
176.241.187.194 - - [10/Sep/2014:08:12:56 +0100] "GET /wp/ HTTP/1.1" 200 45859 "http://theurbanpenguin.co
m/wp/" "Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 6.1; Trident/4.0; SLCC2; .NET CLR 2.0.50727; .NET C
LR 3.5.30729; .NET CLR 3.0.30729; Media Center PC 6.0; .NET4.0C; .NET4.0E; InfoPath.3)"
46.16.40.252 - - [10/Sep/2014:08:13:39 +0100] "GET /browserconfig.xml HTTP/1.1" 404 1319 "-" "Mozilla/5.0
 (Windows NT 6.3; WOW64; Trident/7.0; rv:11.0) like Gecko"
66.249.64.84 - - [10/Sep/2014:08:14:34 +0100] "GET / HTTP/1.1" 200 1608 "-" "DoCoMo/2.0 N905i(c100;TB;W24
H16) (compatible; Googlebot-Mobile/2.1; +http://www.google.com/bot.html)"
195.2.240.101 - - [10/Sep/2014:08:14:43 +0100] "GET /wp/?p=873 HTTP/1.1" 200 16691 "-" "Mozilla/5.0 (Wind
ows NT 6.1; WOW64; rv:26.1) Gecko/20100101 Firefox/26.0"
193.171.130.127 - - [10/Sep/2014:08:15:11 +0100] "GET /wp/wp-content/uploads/2014/05/cropped-wp3.png HTTP
/1.1" 200 65326 "http://theurbanpenguin.com/wp/?p=3121" "Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/
537.36 (KHTML, like Gecko) Chrome/37.0.2062.103 Safari/537.36"
193.171.130.127 - - [10/Sep/2014:08:15:11 +0100] "GET /wp/?p=3121 HTTP/1.1" 200 16813 "http://www.google.
at/url?sa=t&rct=j&q=raspberry%20pi%20dns%20server&source=web&cd=4&sqi=2&ved=0CDUQFjAD&url=http%3A%2F%2Fth
eurbanpenguin.com%2Fwp%2F%3Fp%3D3121&ei=SvoPVObRAcPlatKOgOgM&usg=AFQjCNGhRMN_6hDLlz-_SV2n4KZIP3M6wA&cad=r
ja" "Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/37.0.2062.103 Safa
ri/537.36"
193.171.130.127 - - [10/Sep/2014:08:15:11 +0100] "GET /wp/wp-content/themes/twentytwelve/style.css?ver=3.
9.1 HTTP/1.1" 200 36400 "http://theurbanpenguin.com/wp/?p=3121" "Mozilla/5.0 (Win
```

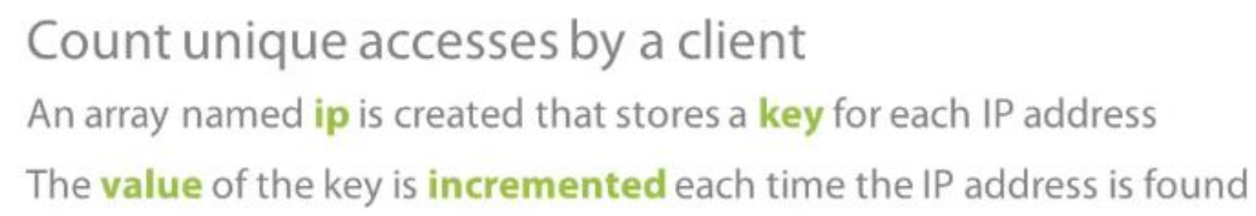


```
user@trusty:~$ awk '$9 == 404 { print $0 } ' access.log
```

ABOVE COMMAND FILTER BY 404 AND PRINT FULL LINE



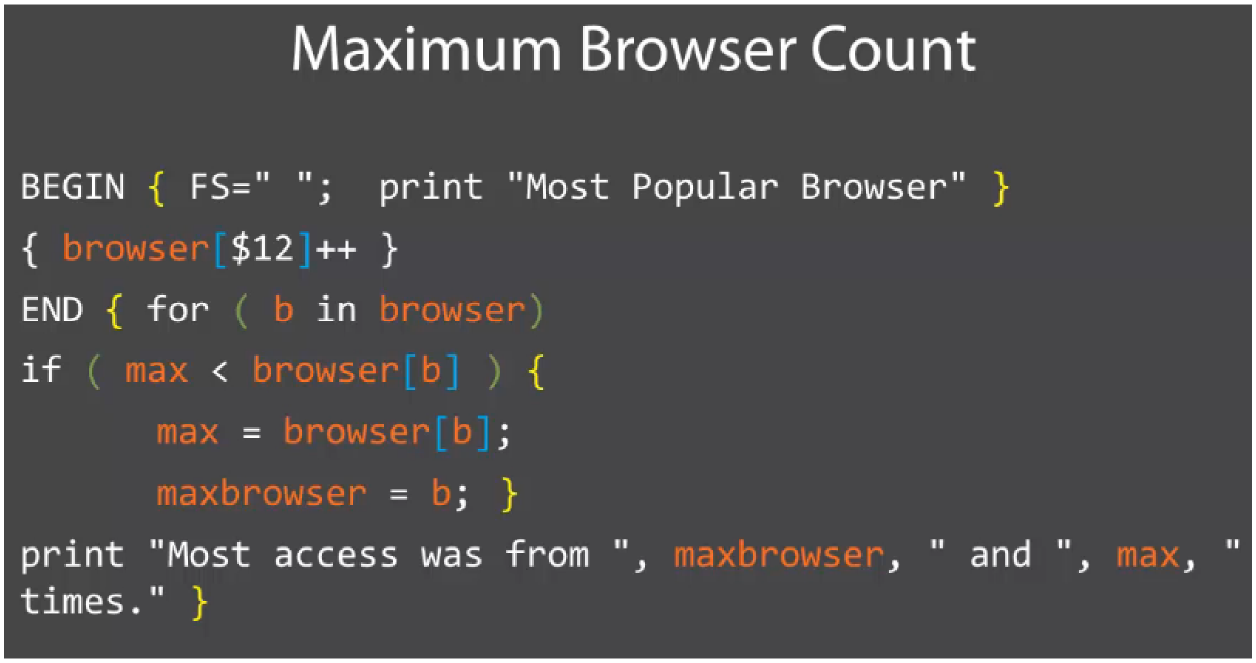
```
$ cat count.awk
BEGIN { FS=" ";print "Log access" }
{ ip[$1]++ }
END { for (i in ip)
print i, " has accessed ", ip[i], " times."
}
 Example:    ip[192.168.0.1]3
```



Count unique accesses by a client

An array named **ip** is created that stores a **key** for each IP address

The **value** of the key is **incremented** each time the IP address is found



Maximum Browser Count

```
BEGIN { FS=" ";  print "Most Popular Browser" }
{ browser[$12]++ }
END { for ( b in browser)
if ( max < browser[b] ) {
       max = browser[b];
       maxbrowser = b;  }
print "Most access was from ", maxbrowser, " and ", max, "
times." }
```