


awk

▼ Type	 Lecture
📅 Date	@February 9, 2022
☰ Lecture #	?
🔗 Lecture URL	
🔗 Notion URL	https://21f1003586.notion.site/awk-14100109861043189e98f31e88705e6d
# Week #	6

awk

A language for processing fields and records

Introduction

- It is a programming language
- **awk** is an abbreviation of the 3 people who developed it
 - **A**ho
 - **W**einberger

- Kernighan
- It is a part of POSIX, IEEE 1003.1-2008
- Variants of `awk`
 - `nawk`
 - `gawk`
 - `mawk`
 - ...
- `gawk` contains features that extend POSIX

Execution model

- Input stream is a set of records
 - Eg. using `"\n"` as a record separator, lines are recorded
- Each record is a sequence of fields
 - Eg. using `" "` as a field separator, words are fields
- Splitting of records to fields is done automatically
- Each codeblock executes on one record at a time, as matched by the pattern of that block

usage

- Single line at the command line

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

- Script interpreted by awk

```
./myscript.awk /etc/passwd
```

```
myscript.awk
```



```
#!/usr/bin/gawk -f
BEGIN {
    FS=":"
}
{
    print $1
}
```

Example #1

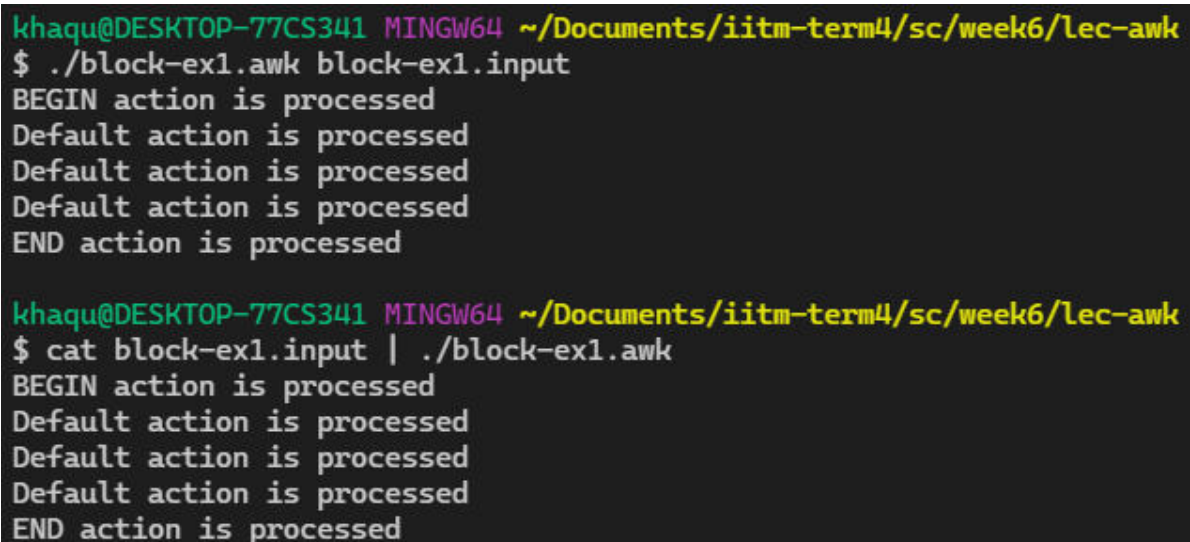
Block example

It does one print statement to illustrate how many times each codeblock is being processed

```
#!/usr/bin/gawk -f
BEGIN {
    print "BEGIN action is processed";
}
{
    print "Default action is processed";
}
END {
    print "END action is processed";
}
```

```
line-1 word1a word1b word1b
line-2 word2a word2b word2b
line-3 word3a word3b word3c
```

Output



The screenshot shows a terminal window with the following commands and output:

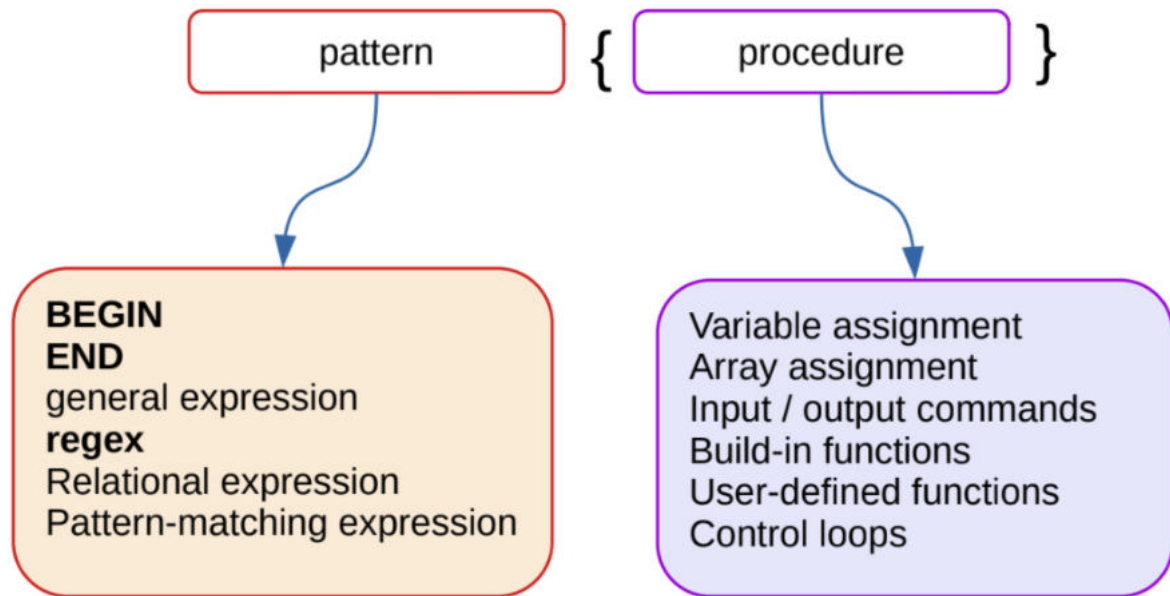
```
khaqu@DESKTOP-77CS341 MINGW64 ~/Documents/iitm-term4/sc/week6/lec-awk
$ ./block-ex1.awk block-ex1.input
BEGIN action is processed
Default action is processed
Default action is processed
Default action is processed
END action is processed

khaqu@DESKTOP-77CS341 MINGW64 ~/Documents/iitm-term4/sc/week6/lec-awk
$ cat block-ex1.input | ./block-ex1.awk
BEGIN action is processed
Default action is processed
Default action is processed
Default action is processed
END action is processed
```

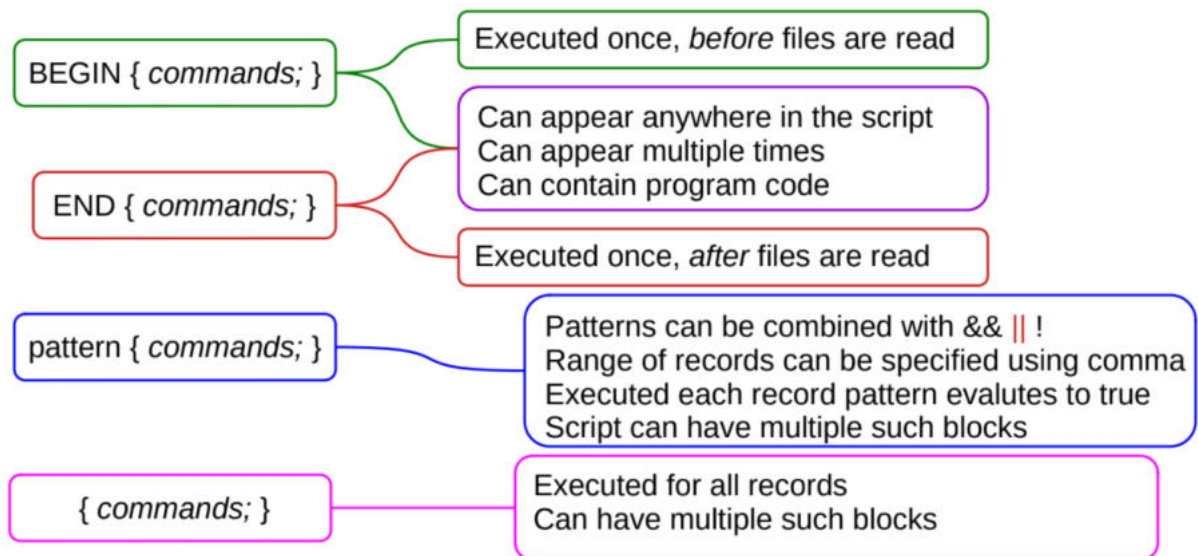
Built-in variables

ARGC	Number of arguments supplied on the command line (except those that came with -f & -v options)
ARGV	Array of command line arguments supplied; indexed from 0 to ARGC-1
ENVIRON	Associative array of environment variables
FILENAME	Current filename being processed
FNR	Number of the current record, relative to the current file
FS	Field separator, can use regex
NF	Number of fields in the current record
NR	Number of the current record
OFMT	Output format for numbers
OFS	Output fields separator
ORS	Output record separator
RS	Record separator
RLENGTH	Length of string matched by match() function
RSTART	First position in the string matched by match() function
SUBSEP	Separator character for array subscripts
\$0	Entire input record
\$n	n th field in the current record

awk scripts



execution



operators

Assignment	= += -= *= /= %= ^= **=
Logical	&&
Algebraic	+ - * / % ^ **
Relational	> <= > >= != ==

<code>expr ? a : b</code>	Conditional expression
<code>a in array</code>	Array membership
<code>a ~ /regex/</code>	Regular expression match
<code>a !~ /regex/</code>	Negation of regular expression match
<code>++</code>	Increment, both prefix and postfix
<code>--</code>	decrement, both prefix and postfix
<code>\$</code>	Field reference
	Blank is for concatenation

Functions and Commands

Arithmetic	<code>atan2 cos exp int log rand sin sqrt srand</code>
String	<code>asort asorti gsub index length match split sprintf strtonum sub substr tolower toupper</code>
Control Flow	<code>break continue do while exit for if else return</code>
Input / Output	<code>close fflush getline next nextline print printf</code>
Programming	<code>extension delete function system</code>
bit-wise	<code>and compl lshift or rshift xor</code>

```
#!/usr/bin/gawk -f
BEGIN {
    print "BEGIN action is processed";
}
{
    print "record: " $0;
    print "processing record number: " FNR;
    print "number of fields in the current record: " NF;
```



```

}
END {
    print "END action is processed";
}

```

```

line-1 word1a word1b word1b
line-2 word2a word2b
line-3 word3a

```

```

khaqu@DESKTOP-77CS341 MINGW64 ~/Documents/iitm-term4/sc/week6/lec-awk
$ ./block-ex2.awk block-ex2.input
BEGIN action is processed
record: line-1 word1a word1b word1b
processing record number: 1
number of fields in the current record: 4
record: line-2 word2a word2b
processing record number: 2
number of fields in the current record: 3
record: line-3 word3a
processing record number: 3
number of fields in the current record: 2
END action is processed

```

```

#!/usr/bin/gawk -f
BEGIN {
    print "BEGIN action is processed";
}
/[[alpha:]]/ {
    print "alpha record: " FNR ":" $0;
    print "number of fields in the current record: " NF;
}
/[[alnum:]]/ {
    print "alnum record: " FNR ":" $0;
    print "number of fields in the current record: " NF;
}
/[[digit:]]/ {
    print "digit record: " FNR ":" $0;
    print "number of fields in the current record: " NF;
}
END {
    print "END action is processed";
}

```

```

hello world welcome to awk
mm22b901 name place
600036 mm23b902 name
04422574770 231

```

Output

```
khaqu@DESKTOP-77CS341 MINGW64 ~/Documents/iitm-term4/sc/week6/lec-awk
$ ./block-ex3.awk block-ex3.input
BEGIN action is processed
alpha record: 1:hello world welcome to awk
number of fields in the current record: 5
alnum record: 1:hello world welcome to awk
number of fields in the current record: 5
alpha record: 2:mm22b901 name place
number of fields in the current record: 3
alnum record: 2:mm22b901 name place
number of fields in the current record: 3
digit record: 2:mm22b901 name place
number of fields in the current record: 3
alpha record: 3:600036 mm23b902 name
number of fields in the current record: 3
alnum record: 3:600036 mm23b902 name
number of fields in the current record: 3
digit record: 3:600036 mm23b902 name
number of fields in the current record: 3
alnum record: 4:04422574770 231
number of fields in the current record: 2
digit record: 4:04422574770 231
number of fields in the current record: 2
END action is processed
```

To match only the first field, instead of the entire record

```
#!/usr/bin/gawk -f
BEGIN {
    print "BEGIN action is processed";
}
$1 ~ /^[[:alpha:]]/ {
    print "alpha record: " FNR ":" $0;
    print "number of fields in the current record: " NF;
}
$1 ~ /^[[:alnum:]]/ {
    print "alnum record: " FNR ":" $0;
    print "number of fields in the current record: " NF;
}
$1 ~ /^[[:digit:]]/ {
    print "digit record: " FNR ":" $0;
    print "number of fields in the current record: " NF;
}
END {
    print "END action is processed";
}
```


Input is the same as the previous

Output

```
khaqu@DESKTOP-77CS341 MINGW64 ~/Documents/iitm-term4/sc/week6/lec-awk
$ ./block-ex4.awk block-ex3.input
BEGIN action is processed
alpha record: 1:hello world welcome to awk
number of fields in the current record: 5
alnum record: 1:hello world welcome to awk
number of fields in the current record: 5
alpha record: 2:mm22b901 name place
number of fields in the current record: 3
alnum record: 2:mm22b901 name place
number of fields in the current record: 3
digit record: 2:mm22b901 name place
number of fields in the current record: 3
alnum record: 3:600036 mm23b902 name
number of fields in the current record: 3
digit record: 3:600036 mm23b902 name
number of fields in the current record: 3
alnum record: 4:04422574770 231
number of fields in the current record: 2
digit record: 4:04422574770 231
number of fields in the current record: 2
END action is processed
```

In the following example, we don't do any pattern matching or comparison

Instead, we look at the number of fields in any particular (or arbitrary) record

```
#!/usr/bin/gawk -f
BEGIN {
    print "BEGIN action is processed";
    FS="[ .;:-]";
}
NF > 2 {
    print "acceptable record:" FNR ":" $0;
    print "number of fields in the current record:" NF;
}
NF <= 2 {
    print "not acceptable record:" FNR ":" $0;
    print "number of fields in the current record:" NF;
}
END {
    print "END action is processed";
}
```

Input

```
hello:world:welcome to awk
mm22b901;name;place
600036-mm23b902-name
04422574770 231 12345
gphani@iitm.ac.in
```

Output

```
khaqu@DESKTOP-77CS341 MINGW64 ~/Documents/iitm-term4/sc/week6/lec-awk
$ ./block-ex5.awk block-ex5.input
BEGIN action is processed
acceptable record:1:hello:world:welcome to awk
number of fields in the current record:5
acceptable record:2:mm22b901;name;place
number of fields in the current record:3
acceptable record:3:600036-mm23b902-name
number of fields in the current record:3
acceptable record:4:04422574770 231 12345
number of fields in the current record:3
acceptable record:5:gphani@iitm.ac.in
number of fields in the current record:3
END action is processed
```

We can modify the FS (field separator) in the above script to not allow " " (space) and "." (dot) as the field separator

The result will change

Modified code (Notice line #4)

```
BEGIN {
    print "BEGIN action is processed";
    FS="[:;-]";
}
NF > 2 {
    print "acceptable record:" FNR ":" $0;
    print "number of fields in the current record:" NF;
}
NF <= 2 {
    print "not acceptable record:" FNR ":" $0;
    print "number of fields in the current record:" NF;
}
END {
    print "END action is processed";
}
```

Input is the same as previous unmodified one

Output

```
khaqu@DESKTOP-77CS341 MINGW64 ~/Documents/iitm-term4/sc/week6/lec-awk
$ ./block-ex5.awk block-ex5.input
BEGIN action is processed
acceptable record:1:hello:world:welcome to awk
number of fields in the current record:3
acceptable record:2:mm22b901;name;place
number of fields in the current record:3
acceptable record:3:600036-mm23b902-name
number of fields in the current record:3
not acceptable record:4:04422574770 231 12345
number of fields in the current record:1
not acceptable record:5:gphani@iitm.ac.in
number of fields in the current record:1
END action is processed
```

Arrays

- Associative arrays
- Sparse storage
- Index need not be integer
- `arr[index]=value`
- `for (var in arr)`
- `delete arr[index]`

Loops

```
for (a in array)
{
    print a
}
```

```
if (a > b)
{
    print a
}
```

```
for (i=1;i<n;i++)
{
    print i
}
```

```
while (a < n)
{
    print a
}
```

```
do
{
    print a
} while (a < n)
```

awk program to calculate the total fee, avg fee and the roll numbers (and their fee) of those who paid less than a threshold from a garbled data file

block-ex6.awk

```
#!/usr/bin/gawk -f
BEGIN {
    print "Report of fee paid:";
    totfee=0;
    FS=" ";
}
{
    # print $0;
    if ($1 ~ /^[[:alpha:]]{2}[[:digit:]]{2}[[:alpha:]]{3}+$/) {
        roll=$1;
        fee=$2;
        rf[roll]=fee;
        totfee += fee;
        print roll " paid " fee;
    }
}
END {
    cutoff=21500;
    print "List of students who paid less than " cutoff;
    ns=0;
    for (r in rf)
    {
        ns++;
        if(rf[r] < cutoff) print "check ", r, " paid only " rf[r];
    }
    avg = totfee/ns;
    print "Total fee collected:" totfee;
    print "Average fee collected:" avg;
}
```

Input

block-ex6.input

```
This file contains comments and numbers
Lines containing roll number and the fee paid need to be picked up
mm22b901 21000
Sum of the total fee paid needs to be printed at the end
Also a list of students, who paid how much
mm22b902 21000
Input text may contain some commends in between
mm22b906      21800
mm22b903 21500 paid through DD
mm22b905      22000
updated as on Jan 26, 2022 by Phani
```

Output

```
khaqu@DESKTOP-77CS341 MINGW64 ~/Documents/iitm-term4/sc/week6/lec-awk
$ ./block-ex6.awk block-ex6.input
Report of fee paid:
mm22b901 paid 21000
mm22b902 paid 21000
mm22b906 paid 21800
mm22b903 paid 21500
mm22b905 paid 22000
List of students who paid less than 21500
check mm22b901 paid only 21000
check mm22b902 paid only 21000
Total fee collected:107300
Average fee collected:21460
```

functions

```
cat infile | awk -f mylib -f myscript.awk
```

mylib
<pre>function myfunc1() { printf "%s\n", \$1 } function myfunc2(a) { return a*rand() }</pre>

myscript.awk
<pre>BEGIN { a=1 } { myfunc1() b = myfunc2(a) print b }</pre>

```
func-lib.awk
```

```
function myfunc1() {
    printf "%f\n", 2*$1;
}
function myfunc2(a) {
    # Adding zero to 'a' here just to store the value in 'b' as an integer
    b=a+0;
    return sin(b);
}
```

```
func-example.awk
```

```
BEGIN {
    FS=":";
    print "----- BEGIN -----";
```

```

    c=atan2(1,1);
    print "c=" c;
    print "-----";
}
{
    print $0;
    myfunc1();
    a=$1;
    b=myfunc2(a);
    print "b=" b;
}
END {
    print "----- ADD -----";
    d=myfunc2(c);
    print "d=" d;
    print "-----";
}

```

Output

```

khaqu@DESKTOP-77CS341 MINGW64 ~/Documents/iitm-term4/sc/week6/lec-awk
$ echo "12.5" | gawk -f func-lib.awk -f func-example.awk
----- BEGIN -----
c=0.785398
-----
12.5
25.000000
b=-0.0663219
----- ADD -----
d=0.707107
-----

```

```

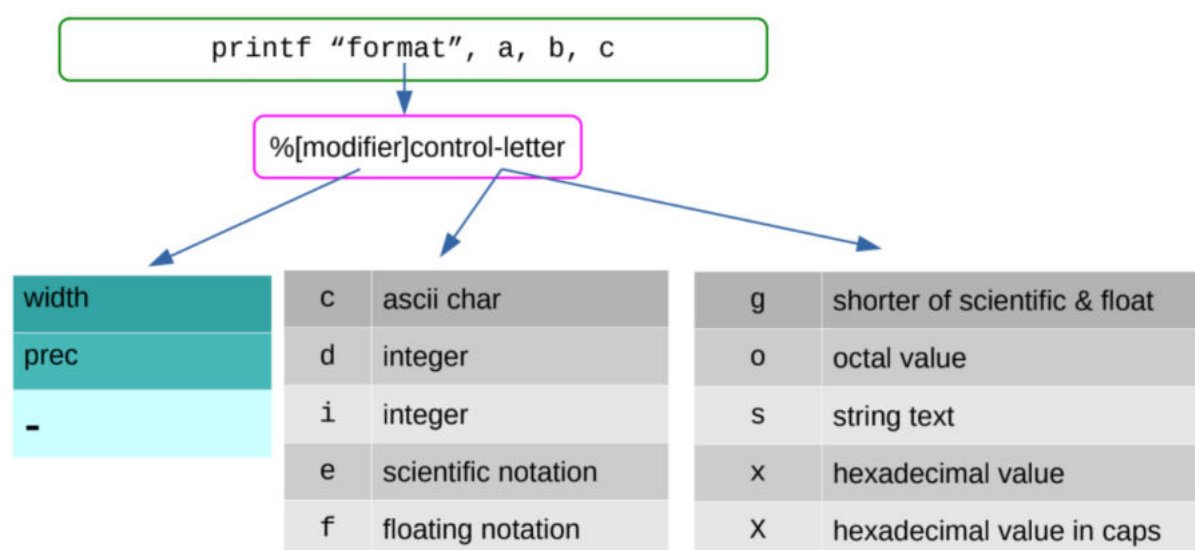
khaqu@DESKTOP-77CS341 MINGW64 ~/Documents/iitm-term4/sc/week6/lec-awk
$ echo "" | gawk -f func-lib.awk -f func-example.awk
----- BEGIN -----
c=0.785398
-----
0.000000
b=0
----- ADD -----
d=0.707107
-----

```



```
khaqu@DESKTOP-77CS341 MINGW64 ~/Documents/iitm-term4/sc/week6/lec-awk
$ touch xaa; cat xaa | gawk -f func-lib.awk -f func-example.awk
----- BEGIN -----
c=0.785398
-----
----- ADD -----
d=0.707107
-----
```

Pretty printing



bash + **awk**

- Including awk inside shell script
- heredoc feature
- Use with other shell scripts on command line using pipe

Examples

The following code creates 3 columns of 10 rows filled with random numbers

We also need to pass an input string for this code

But as we do not have a body for the action block, empty string suffices

rsheet-create.awk

```
#!/usr/bin/gawk -f
BEGIN {
    nl = 10;
    nc = 3;
    for(j=0; j<nl; j++) {
```

```

        for(i=0; i<nc; i++) {
            printf("%f ", rand());
        }
        printf("\n");
    }
}
{}
END {
    print nl " lines with " nc " columns of random numbers created";
}

```

Output

```

khaqu@DESKTOP-77CS341 MINGW64 ~/Documents/iitm-term4/sc/week6/lec-awk
$ echo " " | ./rsheet-create.awk
0.924046 0.593909 0.306394
0.578941 0.740133 0.786926
0.436370 0.332195 0.778880
0.100887 0.785084 0.835159
0.761209 0.496077 0.426298
0.945798 0.821802 0.709269
0.157828 0.119752 0.909685
0.868084 0.449256 0.705432
0.399686 0.645049 0.696163
0.300211 0.591664 0.956569
10 lines with 3 columns of random numbers created

```

Now, we can modify the above code to generate 2 columns of 2 million rows filled with random numbers

We will pass the empty file `xaa` we created earlier, and also use the command `time` to calculate the time taken to execute

```

#!/usr/bin/gawk -f
BEGIN {
    nl = 2000000;
    nc = 2;
    for(j=0; j<nl; j++) {
        for(i=0; i<nc; i++) {
            printf("%f ", rand());
        }
        printf("\n");
    }
}
{}
END {
    # print nl " lines with " nc " columns of random numbers created";
}

```

Output

```
khaqu@DESKTOP-77CS341 MINGW64 ~/Documents/iitm-term4/sc/week6/lec-awk
$ time ./rsheet-create.awk xaa > rsheet-data.txt

real    0m17.544s
user    0m17.500s
sys     0m0.078s
```

Welp, it took a long time on my 🍌 laptop

Now, we should process these numbers and create 2 more rows after that which would contain the product and the sum of these numbers

rsheet-process.awk

```
#!/usr/bin/gawk -f
BEGIN {
    OFS=" ";
    FS=" ";
}
{
    prod = $1*$2;
    sum = $1+$2;
    printf("%f %f %f %f\n", $1, $2, prod, sum);
}
END {}
```

Input is the `rsheet-data.txt` file generated earlier

Output

```
khaqu@DESKTOP-77CS341 MINGW64 ~/Documents/iitm-term4/sc/week6/lec-awk
$ time ./rsheet-process.awk rsheet-data.txt > rsheet-pdata.txt

real    0m53.066s
user    0m52.671s
sys     0m0.077s
```

An awk program to know the IP addresses of all the clients that connected to our web server (Apache) in the last 5 days

Server log file [76.6MB]: <https://firebasestorage.googleapis.com/v0/b/fb-sandbox-25.appspot.com/o/access-full.log?alt=media&token=b1e22c89-5f85-4e12-8087-dfe50fa0e49d>

To get the IP addresses

Type the following code on the bash terminal

```
awk 'BEGIN {FS=" "}{print $1}' access-head.log
```

```
khaqu@DESKTOP-77CS341 MINGW64 ~/Documents/iitm-term4/sc/week6/lec-awk
$ awk 'BEGIN {FS=" "}{print $1}' access-head.log
103.47.219.249
54.209.123.136
54.209.123.136
54.209.123.136
54.209.123.136
54.209.123.136
54.209.123.136
54.209.123.136
54.209.123.136
54.209.123.136
```

```
awk 'BEGIN {FS=" "}{print $1}' access-tail.log
```

```
khaqu@DESKTOP-77CS341 MINGW64 ~/Documents/iitm-term4/sc/week6/lec-awk
$ awk 'BEGIN {FS=" "}{print $1}' access-tail.log
52.203.104.35
52.203.104.35
52.203.104.35
52.203.104.35
52.203.104.35
52.203.104.35
52.203.104.35
52.203.104.35
52.203.104.35
52.203.104.35
```

Here, `access-head.log` and `access-tail.log` are the top 10 and bottom 10 lines of the file `access-full.log` respectively

Similarly, to get the date

Type the following code on the bash terminal

```
awk 'BEGIN {FS=" "}{d=substr($4,2,11); print d}' access-head.log
```

```
khaqu@DESKTOP-77CS341 MINGW64 ~/Documents/iitm-term4/sc/week6/lec-awk
$ awk 'BEGIN {FS=" "}{d=substr($4,2,11); print d}' access-head.log
27/Jan/2022
27/Jan/2022
27/Jan/2022
27/Jan/2022
27/Jan/2022
27/Jan/2022
27/Jan/2022
27/Jan/2022
27/Jan/2022
27/Jan/2022
```

```
awk 'BEGIN {FS=" "}{d=substr($4,2,11); print d}' access-tail.log
```

```
khaqu@DESKTOP-77CS341 MINGW64 ~/Documents/iitm-term4/sc/week6/lec-awk
$ awk 'BEGIN {FS=" "}{d=substr($4,2,11); print d}' access-tail.log
14/Jan/2022
14/Jan/2022
14/Jan/2022
14/Jan/2022
14/Jan/2022
14/Jan/2022
14/Jan/2022
14/Jan/2022
14/Jan/2022
14/Jan/2022
```

Now, combine the above 2 scripts (one from the IP address one, one from the date one) to get the IP address and the date on which they connected to the server

```
awk 'BEGIN {FS=" "}{d=substr($4,2,11); print d, $1}' access-head.log
```

```
khaqu@DESKTOP-77CS341 MINGW64 ~/Documents/iitm-term4/sc/week6/lec-awk
$ awk 'BEGIN {FS=" "}{d=substr($4,2,11); print d, $1}' access-head.log
27/Jan/2022 103.47.219.249
27/Jan/2022 54.209.123.136
27/Jan/2022 54.209.123.136
27/Jan/2022 54.209.123.136
27/Jan/2022 54.209.123.136
27/Jan/2022 54.209.123.136
27/Jan/2022 54.209.123.136
27/Jan/2022 54.209.123.136
27/Jan/2022 54.209.123.136
27/Jan/2022 54.209.123.136
```

```
awk 'BEGIN {FS=" "}{d=substr($4,2,11); print d, $1}' access-tail.log
```

```
khaqu@DESKTOP-77CS341 MINGW64 ~/Documents/iitm-term4/sc/week6/lec-awk
$ awk 'BEGIN {FS=" "}{d=substr($4,2,11); print d, $1}' access-tail.log
14/Jan/2022 52.203.104.35
14/Jan/2022 52.203.104.35
14/Jan/2022 52.203.104.35
14/Jan/2022 52.203.104.35
14/Jan/2022 52.203.104.35
14/Jan/2022 52.203.104.35
14/Jan/2022 52.203.104.35
14/Jan/2022 52.203.104.35
14/Jan/2022 52.203.104.35
14/Jan/2022 52.203.104.35
```

Now, run it on the entire log file and `time` it too. Also save it to a file named `date-ip.txt`

```
time awk 'BEGIN {FS=" "}{d=substr($4,2,11); print d, $1}' access-full.log > date-ip.txt
```

```
khaqu@DESKTOP-77CS341 MINGW64 ~/Documents/iitm-term4/sc/week6/lec-awk
$ time awk 'BEGIN {FS=" "}{d=substr($4,2,11); print d, $1}' access-full.log > date-ip.txt

real    0m0.688s
user    0m0.625s
sys     0m0.031s
```

Statistics time 😊

To calculate the number of connections per day from the log file

To get the datestring for the last n days


```
date --date="<n> days ago" +%d/%m/%Y
```

NOTE: Replace the `<n>` with the actual number

Example

```
khaqu@DESKTOP-77CS341 MINGW64 ~/Documents/iitm-term4/sc/week6/lec-awk
$ date --date="5 days ago" +%d/%m/%Y
05/02/2022

khaqu@DESKTOP-77CS341 MINGW64 ~/Documents/iitm-term4/sc/week6/lec-awk
$ date --date="2 days ago" +%d/%m/%Y
08/02/2022

khaqu@DESKTOP-77CS341 MINGW64 ~/Documents/iitm-term4/sc/week6/lec-awk
$ date --date="3 days ago" +%d/%m/%Y
07/02/2022
```

Code

apache-log-ex1.awk

```
#!/usr/bin/gawk -f
BEGIN {
    ndays=15;
    dformat="+%d/%b/%Y";
    for(i=0; i<ndays; i++) {
        cmdstr=sprintf("date --date=\"%d days ago\" %s", i, dformat);
        cmdstr | getline mydate;
        dates[i]=mydate;
    }
    dstring = "";
    for (i in dates) {
        dstring = dstring " " dates[i];
    }
    print "date string=" dstring;
}
{
    ldate=substr($4,2,11);
    w = match(dstring,ldate);
    if(w != 0) {
        # print ldate " " $1 " " $7;
        print ldate " " $1;
        ipcount[$1]++;
    }
}
END {
    print "----- IP STATS -----";
    for (j in ipcount) {
        print j " " ipcount[j];
    }
}
```

```
}  
}
```

Input

`access-head.log` or `access-tail.log` or `access-full.log`

Output

```
khaqu@DESKTOP-77CS341 MINGW64 ~/Documents/iitm-term4/sc/week6/lec-awk  
$ ./apache-log-ex1.awk access-head.log  
date string= 10/Feb/2022 09/Feb/2022 08/Feb/2022 07/Feb/2022 06/Feb/2022  
8/Jan/2022 27/Jan/2022  
27/Jan/2022 103.47.219.249  
27/Jan/2022 54.209.123.136  
27/Jan/2022 54.209.123.136  
27/Jan/2022 54.209.123.136  
27/Jan/2022 54.209.123.136  
27/Jan/2022 54.209.123.136  
27/Jan/2022 54.209.123.136  
27/Jan/2022 54.209.123.136  
27/Jan/2022 54.209.123.136  
27/Jan/2022 54.209.123.136  
----- IP STATS -----  
103.47.219.249 1  
54.209.123.136 9
```

DNS lookup utility

Command: `dig`

Usage: `dig -x <ip-address>`

Example

```
kashif@zen:~$ dig -x 54.209.123.136  
  
; <<> DiG 9.16.1-Ubuntu <<> -x 54.209.123.136  
;; global options: +cmd  
;; Got answer:  
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 46107  
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1  
  
;; OPT PSEUDOSECTION:  
; EDNS: version: 0, flags:; udp: 65494  
;; QUESTION SECTION:  
;136.123.209.54.in-addr.arpa. IN PTR  
  
;; ANSWER SECTION:  
136.123.209.54.in-addr.arpa. 300 IN PTR ec2-54-209-123-136.compute-1.amazonaws.com.  
  
;; Query time: 88 msec  
;; SERVER: 127.0.0.53#53(127.0.0.53)  
;; WHEN: Fri Feb 11 09:08:39 UTC 2022  
;; MSG SIZE rcvd: 112
```

To get a slimmer output: `dig +noall +answer -x 34.234.167.93`

Example

```
kashif@zen:~$ dig +noall +answer -x 34.234.167.93
93.167.234.34.in-addr.arpa. 300 IN PTR ec2-34-234-167-93.compute-1.amazonaws.com.
```

Previous `awk` code, modified

```
#!/usr/bin/gawk -f
BEGIN {
    ndays=25;
    dformat="+%d/%b/%Y";
    for(i=0; i<ndays; i++) {
        cmdstr=sprintf("date --date=\"%d days ago\" %s", i, dformat);
        cmdstr | getline mydate;
        dates[i]=mydate;
    }
    dstring = "";
    for (i in dates) {
        dstring = dstring " " dates[i];
    }
    print "date string=" dstring;
}
{
    ldate=substr($4,2,11);
    w = match(dstring,ldate);
    if(w != 0) {
        # print ldate " " $1 " " $7;
        # print ldate " " $1;
        ipcount[$1]++;
    }
}
END {
    print "----- IP STATS -----";
    for (j in ipcount) {
        print j " " ipcount[j];
        cmdstr = sprintf("dig +noall +answer -x %s", j);
        cmdstr | getline ipinfo;
        print ipinfo;
    }
}
```