Introduction to AWK

Giovanni Bussi
PhD in Physics and Chemistry of Biological Systems
SISSA, Trieste, Italy

FORTRAN vs C

FORTRAN: born in the 50‘s to translate formulas into programs
evolved through FORTRAN77 - 90 - 03
“close to the math,” but a bit rigid

C: born in the 70’s to write operating systems (UNIX)
evolved with small changes (C89,C99) and one big change (C++)
way more flexible, but less math functionalities

Compiled language vs scripts

Compiled languages (e.g. FORTRAN, C, C++):
source code (e.g. pippo.f90, pippo.c)
compile (e.g. gcc pippo.c -o pippo): commands are "translated"
execute (e.g. ./pippo)
typically: variables declared (robust), very fast exec

Scripting languages (e.g. bash, perl, python, awk)
source code (e.g. pippo.sh)
make it executable (chmod +x ./pippo.sh)
execute (./pippo.sh): commands are "interpreted"
typically: variables not declared (quick), tools to manipulate strings

Bash

Redirection - useful to combine tools

```
# output to a file:
echo “ciao” > file

# input from a file
cat -n < file

# “pipe” a command into another one
echo “ciao” | cat -n
ls -l | grep file
...
```

AWK

Scripting language

C-like syntax

Designed for analysis of text files with columns
(but you can do much more!)

“pattern scanning and processing language”

several clones:
- awk
- gawk
- mawk
- ...

Start

Google “awk tutorial”

http://www.hcs.harvard.edu/~dholland/computers/awk.html

gawk manual:

http://www.gnu.org/software/gawk/manual/gawk.html

One-liners

```
awk '{ print $2, $3 }' < file

awk 'BEGIN{ FS=":"} { print $6}' < file

awk '{ print ($1-32)*(5/9) }'

echo 5 4 | awk '{ print $1 + $2 }'

echo 5 4 | awk '{ print $1 $2 }'

echo 5 4 | awk '{ print $1, $2 }'
```

Longer scripts

Create a file like this:

```
#! /bin/bash
awk '{
   print $2, $3;
}'
```

```
chmod +x ./pippo.sh
```

```
./pippo.sh < file
```

Variables and flow control

```
# average of columns
awk '{
    tot=0; for (i=1; i<=NF; i++) tot += $i;
    print tot/NF;
}'
```

```
# count columns with value larger than 10
# but only for even rows
awk '{
    tot=0;
    for (i=1; i<=NF; i++)
      if($i>10 && NR%2==0) tot++;
    print tot;
}'
```

A note on variables

Every variable is a string
Non-numeric strings are treated as 0

```
# print line and "line+1" (whatever it means)
awk '{
   tot=$0
   print tot,tot+1
}'
```

Predefined variables:
$1, $2, ... : columns
$0 : full row
NR : number of row
NF : number of fields (columns)

A note on algebra

```
awk '{
   x=$1; y=$2;
   print sin(x),cos(x),exp(x);
   print x*x+2;
   print x+y;
   x++; print x; # increases x by one
   x-=1; print x; # decreases x again
   if(x>0 && y>0) print "x and y positive"
   if(x>0 || y>0) print "x or y positive"
}'
```

A note on C-like for

```
for ( initialize ;condition; increment )
{ block}
```

```
# average of odd columns
awk '{
    tot=0; for (i=1; i<=NF; i+=2) tot += $i;
    print tot/NF;
}'
```

Blocks

```
# average of column 1
awk '{
    tot += $1; n += 1;
} END {
   print tot/n;
}'
```

Unnamed block: execute for every line of input
BEGIN block: execute before input starts
END block: execute at the end
/xxx/ blocks: search for regexp xxx (check on google for examples)

Printf

```
# print from column 2 to last column
# on the same row
awk '{
   for (i=2; i<=NF; i++) printf("%s ", $i);
   printf("\n");
}'
```

```
# again but with fixed format
awk '{
   for (i=2; i<=NF; i++) printf("%11.4f ", $i);
   printf("\n");
}'
```

Other

```
# a standalone program that prints number 0..9
awk 'BEGIN{ for(i=0;i<10;i++) print i; }'

# if/then/else
awk '{
   if($1>0 && $1<1){
      c++
   } else if($1<2 && $1>1){
      d++
   } else e++;
}END{
 print c,d,e;
}'
```

Arrays

```
# save every line in an array element
# and write them in reverse order
awk ‘{
  line[NR]=$0;
}END{
  for(i=NR;i>0;i--) print line[i];
}’
```

Arrays are associative

```
# arrays are associative!
awk '{ age[$1]=$2; # column 1 is used as index
}END{
  for(name in age) print age[name];
}'
```

```
# no need to allocate, but can deallocate
awk ‘{ line[$1]=$2; }END{
  for(i in line) printf(“%s ”,line[i]); printf(“\n”);
  delete line[1]; # remove element 1
  for(i in line) printf(“%s ”,line[i]); printf(“\n”);
  delete line;
  for(i in line) printf(“%s ”,line[i]); printf(“\n”);
}’
```

Functions

```
awk '
function f(x,n) {
  return n*x;
}{
  print f($1,NR)
}'
```

```
awk '
function f(x,n, y) {
  y=1; for(i=1;i<=n;i++) y*=x; return y;
}{
  print f($1,NR)
}'
```

Random numbers

```
awk 'BEGIN{
   srand(10); # initialize seed
   for(i=0;i<100;i++){
      r=rand(); # uniformly in (0,1)
      print r;
   }
}'
```

Don't forget bash!

```
# list files larger than 1000 bytes, sorted
ls -l |
  awk '{if($5>1000) print $5,$NF}' |
  sort -n
```

```
# compute average of 1000 random numbers
awk 'BEGIN{
  for(i=0;i<1000;i++) print rand()}' |
awk '{x+=$1;c++}END{print x/c}'
```