

A short AWK introduction

<https://github.com/mschmnet/awk-slides>

© 2019 Manuel Schmidt

Contents

1. What is AWK for?
2. Basics
3. More advanced features

1. **What is AWK for?**
2. Basics
3. More advanced features

Why should I use AWK for?

“ I need to process an input file by splitting it up in
records and **fields** ”

How are files split into *records* and *fields*?

- By default the input record separator is `\n` and the default field separator is any number of **spaces** and/or **tabs** and/or **new line** characters.
- The defaults can be changed with the special variables `RS` (*Record Separator*) and `FS` (*Field Separator*)
- The *field separator* can be an extended regular expression

What is the AWK's input?

- It can be *standard input*
- It can be one or more file
- It can be mixed (one or more files + *standard input*)

Multiple input files

- You can specify multiple files: `awk '{print $1}' input.txt input-2.txt`
- You can mix files with *standard input*: `cat input.txt | awk '{print $1}' - input-2.txt`
- You can distinguish between first and subsequent files comparing special variables `NR` and `FNR`: `awk 'NR == FNR{print "From 1st file: " $0}NR != FNR{print "From 2nd file: " $0}' input.txt input-2.txt`

1. What is AWK for?
2. **Basics**
3. More advanced features

Where to put your awk coke

- Inline: `awk '{print $1}' input.txt`
- In a separate file

```
#!/usr/bin/awk -f  
  
{  
    print $1  
}
```

- You can execute it with `awk -f ./script.awk input.txt`
- Or you can assign execution permission to the file and execute it directly: `chmod +x script.awk;`
`./script.awk < input.txt`

Main blocks

- `BEGIN{ }` : This block of code is executed *before* the first record is processed
- `END{ }` : This block is executed *after* the last record has been processed
- A `{ }` block is execute on every *record*.
- There are other blocks like `BEGINFILE{ }`

Example (input: temperature-salamanca.csv)

"Salamanca Aeropuerto"

Actualizado: martes, 16 julio 2019 a las 18:42 hora oficial

"Fecha y hora oficial","Temperatura (°C)",[...],"Humedad (%)"

"16/07/2019 18:00","32.0",[...],"26"

[...]

"15/07/2019 19:00","30.8",[...],"24"

Example (code: temperature.awk)

```
#!/usr/bin/awk -f
```

```
BEGIN{
    FS="\(\\", \"\)\|\"";
}
$2 ~ /^[0-9][0-9]\\/[0-9][0-9]\\/[0-9]{4} [0-9][0-9]:[0-9][0-9]$/ {
    date_time=$2;
    temperature=$3;
    humidity=$11;
    total_temperature+=temperature;
    total_humidity+=humidity;
    n++;
}
END{
    printf "Average temperature: %.2f\n", total_temperature / n;
    printf "Average humidity: %.0f\n", total_humidity / n;
}
```

Example (output)

```
./temperature.awk < temperature-salamanca.csv
```

```
Average temperature: 23.96
```

```
Average humidity: 42
```

Record processing

- To every record a list of pairs applies. Every pair is made up of:
 - A **condition** that checks if following code block must be executed or not
 - If missing, following code block will be executed
 - **Executable code** (within curly braces)
 - If missing, complete record will be printed (`{print $0}`)

Record processing (examples)

- Both, condition and code block, are present: `awk '$1 ~ /one/{print "Number 1"}' < input.txt`
- Condition is missing: `awk '{print "Number " $1}' < input.txt` (same as `'1{print "Number " $1}'`
- Code block is missing: `awk '$1 ~ /one/' < input.txt` same as `$1 ~ /one/{print $0}`
- More than one pair may exist: `awk '0{print "This will never be printed"}1{print "This will always be printed"}' < input.txt`

1. What is AWK for?
2. Basics
3. **More advanced features**
 - 3.1 Arrays

1. What is AWK for?
 2. Basics
 3. More advanced features
- 3.1 **Arrays**

Nature of arrays

- Arrays are associative. Awk doesn't have linked lists.
- Arrays in awk can be looped through with a *for each* syntax, but you don't get insertion order.
- If you want insertion order you have to use an incremental index and use a regular loop with an incremental index (`for(i=1;i<=length(a);i++)`
`{print a[i];}`)

A few examples with arrays (1)

Column 3 of `Average_Daily_Traffic_counts.csv` is a street name. Print the first record for every street (uniq by column)

```
awk -F"," ' !_[$3]++ ' Average_Daily_Traffic_counts.csv
```

A few examples with arrays (2)

Column 3 of `Average_Daily_Traffic_counts.csv` is a street name. Print the number of occurrences.

```
BEGIN{
    FS=",";
    OFS=" ==> "
}
NR >= 1{
    _[$3]++
}
END{
    for(el in _){
        print el, _[el];
    }
}
```

Multidimensional arrays

They aren't really multidimensional. But, who cares? Or maybe you should?

```
a["2019", "Salamanca"]=1342

for(el in a){
  print el, a[el]
  # ??
}
```

