worteks
make IT work, make IT free

PARIS OPEN SOURCE SUMMIT

Learn AWK in 15 minutes

Maxime Besson
info@worteks.com

Vorteks
make IT work, make IT free

(\vɔʁ.tɛks\)

PARIS OPEN SOURCE SUMMIT

Services

Heterogeneous and complex infrastructures, cloud, mail, authentication, security

- Studies, audit and consulting
- Technical expertise
- Technical support
- Training
- R&D

V'Sweet — Collaboration and application portal

V'Opla — Mutualized platform for development

FUSION IAM — Identity and Access Management

Partnership

BlueMind

redhat
READY
BUSINESS PARTNER

Why learn AWK in $(date +%Y)?

- It’s everywhere:

```
# Yes, everywhere:
$ docker run -it --rm alpine awk 'END{print "Hello"}' /dev/null
Hello
```

- It’s powerful and versatile
- It plays nicely with many other command line tools
- This:

  https://adamdrake.com/command-line-tools-can-be-235x-faster-than-your-hadoop-cluster.html

What I'm going to teach you

```
awk -F: '($3<1000){s++} END{print s}'
```

- Don't worry, it looks scarier than it really is !
- If you understand what this does, you can sleep until the next talk !

Structure of an AWK command

```
awk [program arguments] code [input file]
```

- Program arguments change how the interpreter works
- Code is an awk script, usually quote-protected
- Input file is optional, awk will then read the standard input
- Awk writes to standard output
- You can put the script in a file if it gets too big for a one-liner

What AWK does (pseudo-code)

```
foreach line
    if (CONDITION) then
        ACTIONS
    endif
    if (CONDITION) then
        ACTIONS
    endif
    …
endfor
```

What you write (real AWK code)

```
CONDITION {
    ACTIONS
}
CONDITION {
    ACTIONS
}
```

What you write (one-liner)

```
awk ‘CONDITION {ACTIONS} CONDITION {ACTIONS}’
```

AWK conditions

- if CONDITION empty, INSTRUCTIONS are applied to every line
- BEGIN / END : will be run only once
- /regexp/
- (expression) : usually a test for equality
- some others...
- Any logical combination of the above

AWK variables

- AWK allows you to set any variable you like
- AWK defines some variables for you:
  - NR : current „record” (line) number
  - NF : how many „fields” the record has
  - $0 : the current line
  - $1, $2, … : n-th field on the line

  Fields are delimited by tabs and spaces by default, this can be overriden with the -F option
- AWK is great at handling CSV files !

That's all...

- Now you know (almost) all the theory. The rest is in man pages

Examples

```
awk '{print NR}'
```

Examples

```
awk '{print NR}'
awk '{print NR, $0}'
```

Examples

```
awk '{print NR}'
awk '{print NF}'
awk  'END {print NR}'
```

Examples

```
awk '{print NR}'
awk '{print NF}'
awk  'END {print NR}'
awk ' /root/ { print $0 }'
```

Examples

```
awk -F: ' /root/ { print $3 }' /etc/passwd
```

Examples

```
awk -F: ' /root/ { print $3 }' /etc/passwd
awk -F: ' ($3==1000) { print $1 }' /etc/passwd
```

Examples

```
awk -F: ' /root/ { print $3 }' /etc/passwd
awk -F: ' ($3==1000) { print $1 }' /etc/passwd
awk -F: ' ($1==”root”) { print $6 }' /etc/passwd
```

Examples

```
awk -F: ' /root/ { print $3 }' /etc/passwd
awk -F: ' ($3==1000) { print $1 }' /etc/passwd
awk -F: ' ($1==”root”) { print $6 }' /etc/passwd
awk -F: '($6 ~ "^/home"){ c++ } END{print c}'
```

Conclusion

- When should I use awk
  - Shell one liners
  - One shot processing of CSV data
    - '($2 == something) {sum = sum+ $3}'
    - group-by using arrays
- When should I not use awk
  - Internationalization
  - Complex calculations
  - Complex data structures
  - Long scripts in general

worteks

make IT work, make IT free

PARIS OPEN SOURCE SUMMIT

Thanks for your attention

More informations:

info@worteks.com

@worteks_com

linkedin.com/company/worteks