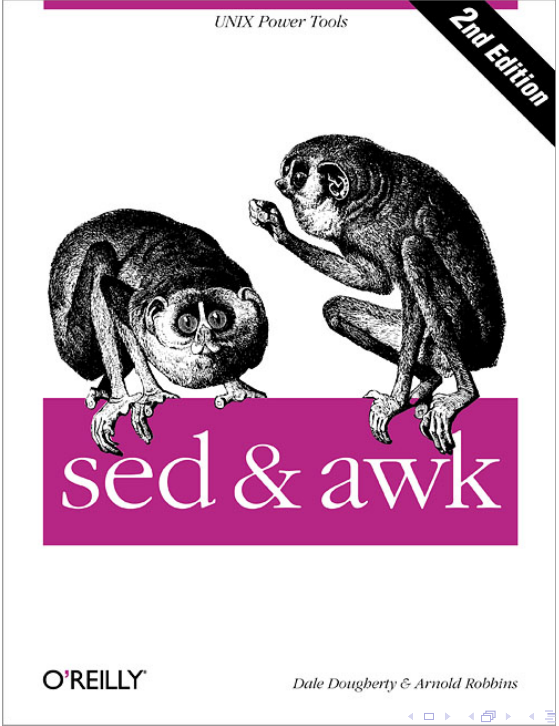
AWK

Seth House <seth@eseth.com>

Ogden Area Linux User Group

2013-03-26

Outline


UNIX Power Tools
2nd Edition
sed & awk
O'REILLY
Dale Dougherty & Arnold Robbins

awk, nawk, gawk

- AWK (language)
- awk 1977 (oawk)
- nawk 1984 (POSIX)
- gawk

Outline

AWK

Structure

```
''condition'' { ''action'' }
```

Fields

- `echo "one two three" | awk '{ print $1, $3 }'`
- `echo "one two three" | awk '{ print $0 }'`

Records

```
echo -e "one two three\nfour five six" | \
    awk '{ print $1, $3 }'
```

Types of blocks

- `BEGIN`
- `END`
- condition

Outline

Conditions

Multiple blocks

```
echo -e "one two three\nfour five six" | \
    awk '
        { print $1, $3 }
        { print $2 }
    '
```

Fields

```
echo -e "one\ntwo\nthree" | \
    awk '$1 == "two" { print $1 }'
```

Equality `$2 == "Sam"`

Comparison `$2 > 1`

Combine conditions `$2 > 1 && $2 < 5` `$2 == "Sam" || $2 == "George"`

Multiple conditions `$2 == "Sam", $3 < 5`

Regex

```
echo -e "alpha\n1\nbeta\n2" | \
    awk '/[a-z]/ { print $0 }'
```

Negation

```
echo -e "alpha\n1\nbeta\n2" | \
    awk '! /[a-z]/ { print $0 }'
```

Matching

```
echo -e "foo\nbar\nbaz\n" | \
    awk '$1 ~ /^ba/ { print $1 }'
```

Outline

Printing output

print

```
echo "one two three" | \
    awk '{
        print "Records"
        print "First: " $1, "Second: " $2,
            "Third: " $3
    }'
```

Field separator

Default: whitespace

```
echo "one,two,three" | \
    awk 'BEGIN { FS="," } { print $1, $3 }'
```

```
echo "one,two,three" | \
    awk -F, '{ print $1, $3 }'
```

Record separator

Default: newline

```
echo -e "one\ntwo\n\nthree\nfour" | \
    awk 'BEGIN {
            RS=""   # blank line
            FS="\n" # newline
        }
        { print $1 }
    '
```

Output field separator

Default: space

```
echo "one two three" | \
    awk 'BEGIN { OFS="," }
        { print $1, $2, $3 }'
```

Output record separator

Default: newline

```
echo -e "one two three\nfour five six" | \
    awk 'BEGIN { OFS=","; ORS=";" }
        { print $1, $2, $3 }'
```

printf

```
printf
```

Output format

Default: “%.6g”

OFMT: Stores the format for numeric output.

Outline

Variables

Number of input fields

```
echo "one two three" | \
    awk '{ print NF }'
```

Number of input records

```
echo -e "one\ntwo\nthree" | \
    awk '{ print NR }'
```

Input filename

```
echo "one" > one
echo "three" > three
echo "two" | awk '{ print FILENAME }' one - three
```

Environment variables

```
WTF="bbq" awk '{ print ENVIRON["WTF"] }'
```

Number of args

```
awk 'END { print ARGC }' file1 file2
```

Array of args

```
awk 'END { for (i in ARGV) print i }' file1 file2
```

Assignment, variable types, counters

- No need to initialize variables

Type casting

- Strings to numbers (arithmetic / counters)
- Numbers to strings (`print`)

Associative arrays

```
times_seen[$1] += 1
```

Builtin functions

Custom functions

```
function add_three (number) {
    return number + 3
}

print add_three(36)
```

Outline

Usage & mechanics

Calling awk

Stdin	`somecmd | awk '{ ... }'`
Input file(s)	`awk '{ ... }' input-file1 input-file2`
Stdin and input files	`somecmd | awk '{ ... }' input-file1 - input-file2`

Writing AWK

Inline	`awk '{ ... }'`
External script	`awk -f myscript.awk`
Shell script (`chmod +x`)	`#!/usr/bin/awk -f`

Readable awk inside another script

```
#!/bin/sh
somecmd | awk '
BEGIN {
    ...
}
/match/ {
    ...
}
END {
    ...
}'
```

Arguments

```
echo "blah" | awk somearg=someval '{ print somearg }'
```

Outline

Real world awk

Committers by number of commits

```
git log --format='%aN <%aE>' | \
    awk '{arr[$0]++}
        END {
            for (i in arr){ print arr[i], i; }
        }' | sort -rn
```

Merged pull requests by date

```
git log --date=relative \
    --pretty="format:%h %ci" \
    --grep "Merge pull request" | \
    awk '{ dc[$2]+=1 }
    END { for (d in dc) print d, dc[d] }' | sort
```

IRC channel stats

```
awk '$2 ~ /\<\w+\>/ {
    file[FILENAME]+=1;
    people[$2]+=1;
    count+=1
}
END {
    print "Avg per day:", count / (ARGC - 1);
    for (i in people) n+=1;
    print "By", n, "people";
    max=0;
    for (i in file) { if (file[i] > max) max=i; };
    print "Busiest day was", max, "with", \
        file[max], "things said";
}' \#utahjs*
```