CS2043 - Unix Tools & Scripting

Cornell University, Spring 2014[1]

Instructor: Bruno Abrahao

February 14, 2014

[1] Slides evolved from previous versions by Hussam Abu-Libdeh and David Slater

Recap: gawk functions

- `exp(x)` : exponential of `x`
- `rand()` : produces a random number between 0 and 1
- `length(x)` : returns the length of x
- `log(x)` : returns the log of x
- `sin(x)` : returns the sin of x
- `int(x)` : returns the integer part of x

Recap: What type of code can I use in gawk?

gawk coding is very similar to programming in c

- `for(i = ini; i <= end; increment i) {code}`
- `if (condition) {code}`

(In both cases the { } can be removed where only one command is executed)

Recap: Example

- `NF` - # of fields in the current line
- `NR` - # of lines read so far

```
gawk '{for (i=1;i<=NF;i++) print $i }' infile
```

Prints all words in a file

- You **cannot** change `NF` or `NR`.

Variables

- gawk handles variable conversion automatically

```
total = 2 + "3" // assigns 5

total++ // total = total + 1

++total // returns current value, then total = total + 1

line = "foo" "bar" // concatenates two strings

line = var "bar" // concatenates the contents of var with bar
```

Variables

Operators

- ++ Add 1 to variable.
- -- Subtract 1 from variable.
- += Assign result of addition.
- -= Assign result of subtraction.
- *= Assign result of multiplication.
- /= Assign result of division.
- %= Assign result of modulo.
- **= Assign result of exponentiation

Another gawk function

- `substr(string, beg[, len])` : Return substring of string at beginning position beg (counting from 1), and the characters that follow to maximum specified length len. If no length is given, use the rest of the string.

gawk examples

```
gawk ' {
    for(i=1;i<=NF;i++){
        for(j=length($i);j>0;j--) {
            char = substr($i,j,1)
            tmp = tmp char
        }
        $i = tmp
        tmp = ""

    } print
} ' infile
```

- What does this do?

Associative Arrays

Abstract data type composed of a collection of `(key,value)` pairs, where each possible key appears at most once in the collection. Supports the following operations:

- the addition of pairs to the collection
- the modification of the values of existing pairs
- the lookup of the value associated with a particular key
- the deletion of pairs to the collection

Other names: *dictionary*, *map*, *hash table*

Associative Arrays

`(key,value)` addition

- Arrays are automatically created and resized
- "associative" means that the index can be any string:

`array["txt"] = value`
`array[50]` is equivalent to `array["50"]`.

Associative Arrays

`(key,value)` modification

```
array["txt"]++
array["txt"]+= $1
array["txt"]+= $1 "bar"
```

Associative Arrays

(key,value) lookup

```
print array["txt"]
array["txt"]= array["txt"] "bar"
```

Associative Arrays

`(key,value)` deletion

```
delete array["txt"]
```

Associative Array functions

```
if (someKey in theArray) {
    code
} else {
    code
}

for (i in theArray)  code
```

Fun

Let's implement `sort | uniq -c` in `awk`!

Associative Array Example

Suppose we have an iou file of the following form:

```
Who owes me what as of today
Name \tab Amount
Name \tab Amount
⋮
```

Lets write a gawk script to add up how much everyone owes us

Associative Array Example

```
gawk '
    BEGIN {FS = "\t" }
    NR > 1 { Names[$1]+=$2 }
    END { for(i in Names) print i " owes me "
    Names[i] " Dollars."}
' ioufile
```

Formatter Printing

```
printf("Hello World\n")

printf("%d\t\%s\n", $5, $9)
```

where

- %d: decimal integer
- %s: string
- \t: tab
- \n: new line

Happy Valentine's! Have a wonderful break!