C2110 *UNIX and Programming*

11th lesson

awk

Petr Kulhánek

kulhanek@chemi.muni.cz

National Centre for Biomolecular Research, Faculty of Science,
Masaryk University, Kamenice 5, CZ-62500 Brno

Test II

Test II

- **Test through ROPOT in IS** (ROPOT = Revision, Opinion Poll and Testing)

Student - ROPOT - C2110 - Test 2c

Time limit - 20 minutes.

Only one set of questions can be built.

Continuously save your answers.

Evaluation can be done only once.

It is allowed and recommended:

- to test commands in the terminal.
- to search in the manual pages, in your notes and presentations of the course.
- when in doubt, ask the teacher.

It is not allowed:

- to communicate with another person except the teacher.

Contents

➢ AWK

- What is AWK?
- Script structure, Process execution
- Block structure
- Variables, Operations with variables
- Conditions
- Loops

AWK

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

AWK is a scripting language designed for **processing text data**, whether in the form of text files or streams. The language uses the **string data types, associative arrays** (indexed by string keys) and **regular expressions**

adaptováno z www.wikipedia.org

Text file analysis

```
54.7332   295.7275   128.4090  -508.1302  -155.6037     0.0000
51.3204   292.3619   176.5980  -494.7423  -164.7991     0.1822
40.6154   273.9238   164.5827  -488.9232  -163.0629     0.3793
52.5044   281.5944   153.4570  -484.6533  -168.5328     0.3528
62.5486   294.2701   155.3607  -483.6872  -169.1747     0.0033
```

```
Potential function:
     ntf     =       2, ntb     =       0, igb     =       5, nsnb    =      25
     ipol    =       0, gbsa    =       0, iesp    =       0
     dielc   =   1.00000, cut     = 999.00000, intdiel =   1.00000
```

Text file analysis

record

field of the record

```
54.7332   295.7275   128.4090   -508.1302   -155.6037   0.0000
51.3204   292.3619   176.5980   -494.7423   -164.7991   0.1822
40.6154   273.9238   164.5827   -488.9232   -163.0629   0.3793
52.5044   281.5944   153.4570   -484.6533   -168.5328   0.3528
62.5486   294.2701   155.3607   -483.6872   -169.1747   0.0033
```

field of the record

record

```
Potential function:
    ntf     =       2, ntb     =       0, igb     =       5, nsnb    =      25
    ipol    =       0, gbsa    =       0, iesp    =       0
    dielc   =   1.00000, cut     = 999.00000, intdiel =   1.00000
```

Text file analysis

```
 54.7332   295.7275   128.4090  -508.1302  -155.6037     0.0000
 51.3204   292.3619   176.5980  -494.7423  -164.7991     0.1822
 40.6154   273.9238   164.5827  -488.9232  -163.0629     0.3793
 52.5044   281.5944   153.4570  -484.6533  -168.5328     0.3528
 62.5486   294.2701   155.3607  -483.6872  -169.1747     0.0033
```

```
Potential function:
     ntf     =       2, ntb     =       0, igb     =       5, nsnb    =      25
     ipol    =       0, gbsa    =       0, iesp    =       0
     dielc   =   1.00000, cut     = 999.00000, intdiel =   1.00000
```

Process of script execution

```
1  BEGIN {
   }
2  {
   }
3  /PATTERN/ {
   }
4  END {
   }
```

- BEGIN block (1) is executed (if it is part of the script) before the file analysis.
  - Record is read from the file. By default, the record is one line of the analyzed file or stream. The record is split into fields. By default, the fields are individual words of the record.
  - Block (2) is executed for the given record.
  - If the record matches the pattern, the block (3) is executed.
  - ... potential execution of other blocks ...
- Block END (4) is executed (if it is included in the script) after analysis of the whole file.

Structure of AWK script

```
BEGIN {

}

{

}

/VZOR/ {

}

END {

}
```

executed before processing the file (optional section)

executed for each record

executed for each record matching the PATERN

any number of sections

executed after processing the file (optional section)

The block is enclosed in curly braces {}.
Program blocks as shown are optional.
Line is set as a record in default setting.

Example

input.txt

```
54.7332   295.7275   128.4090   -508.1302   -155.6037   0.0000
51.3204   292.3619   176.5980   -494.7423   -164.7991   0.1822
40.6154   273.9238   164.5827   -488.9232   -163.0629   0.3793
52.5044   281.5944   153.4570   -484.6533   -168.5328   0.3528
62.5486   294.2701   155.3607   -483.6872   -169.1747   0.0033
```

script.awk

```
{
    print $2;
}
```

one simple block

```
$ awk -f script.awk input.txt
                or
$ awk '{ print $2; }' input.txt
```

```
295.7275
292.3619
273.9238
281.5944
294.2701
```

Block structure

comment starts with character #

```
# This block counts subtotal and analyzes
# value of the fourth column
{
    # this is comment
    i = i + 1;
    f = f + $2; # here i counts subtotal
    printf("Subtotal is %10.3f\n",f);
    if( $3 == 5 ) {
        k = k + $4;
    }
}
```

Commands are placed on separate lines. It is appropriate to end the line with the semicolon despite awk does not require this.
Semicolon must be used when two or more commands are placed on one line.

Variables

Assignment to a variable:

```
A = 10;
B = "this is text"
C = 10.4567;
D = A + C;
```

Value of variable:

```
print A + C;
print B;
```

Must not contain spaces

Differences from BASH

```
A=5

echo $A
```

Value of variable using **$**

Special variables:

NF	number of fields in the current record
NR	index number of current record
FS	field separator, **in default it is space and tabulator**
RS	record separator, **it default it is character for new line \n**
$0	whole record
$1, $2, $3 ...	individual fields of the record

Variables,...

$0	whole record
$1, $2, $3 ...	individual fields of the record

character **$** allows for program access to individual fields of the record

Example:

```
i=3;
print $i;
```

prints the value of third column

Launching of AWK scripts

Processing of text file:

Indirect launch:

```
$ awk -f script.awk input.txt
```

result is printed on the screen

analyzed text file

awk script

language interpreter

Analyzed data can be sent through standard input:

```
$ awk -f script.awk < input.txt

$ cat input.txt | awk -f script.awk
```

Launching of AWK scripts,...

Direct launch:

```
$ ./script.awk input.txt

$ ./script.awk < input.txt

$ cat file.txt | ./script.awk
```

File `script.awk` **must have** set flag x (**executable**) and interpreter (part of the script).

```
#!/usr/bin/awk -f
{
    i += NF;
}
END {
    print "Number of words:", i;
}
```

Exercise

1. In your home directory, create directory awk-data
2. Copy file matice.txt, produkt.log, and rst.out to directory awk-data from directory /home/kulhanek/Documents/C2110/Lesson11.
3. Write a script which will print the second column from the file matice.txt.
4. Write a script which will print the second and fourth column from the file matice.txt

matice = matrix
produkt = product (chemistry)

Mathematical operations

If a variable can be interpreted as an integer, following arithmetic operators can be used:

++ value of the variable is increased by one

```
A++;
```

-- value of the variable is decreased by one

```
A--;
```

+ sums up two values

```
A = 5 + 6;
A = A + 1;
```

- subtracts two values

```
A = 5 - 6;
A = A - 1;
```

***** multiplies two values

```
A = 5 * 6;
A = A * 1;
```

/ divides two values

```
A = 5 / 6;
A = A / 1;
```

+= adds value to variable

```
A += 3;
A += B;
```

-= subtracts value from variable

```
A -= 3;
A -= B;
```

***=** multiplies variable by value

```
A *= 3;
A *= B;
```

/= divides variable by value

```
A /= 3;
A /= B;
```

Command print

Command **print** serves for non-formatted output of strings and numbers.

Syntax:

```
print value1[,] value2[,] ...;
```

if values are separated by a comma, values are separated by a space in the output

Examples:

```
i = 5;
k = 10.456;
j = "value of variable i =";
print j, i;
print "value of variable k =", k;
```

Exercise

1. Write a script which will sum numbers in the second column of the file matice.txt.
2. Write a script which will print the number of lines of the file matice.txt.

   Verify the result by using wc command.
3. Write a script which will print the number of words in the file matice.txt.

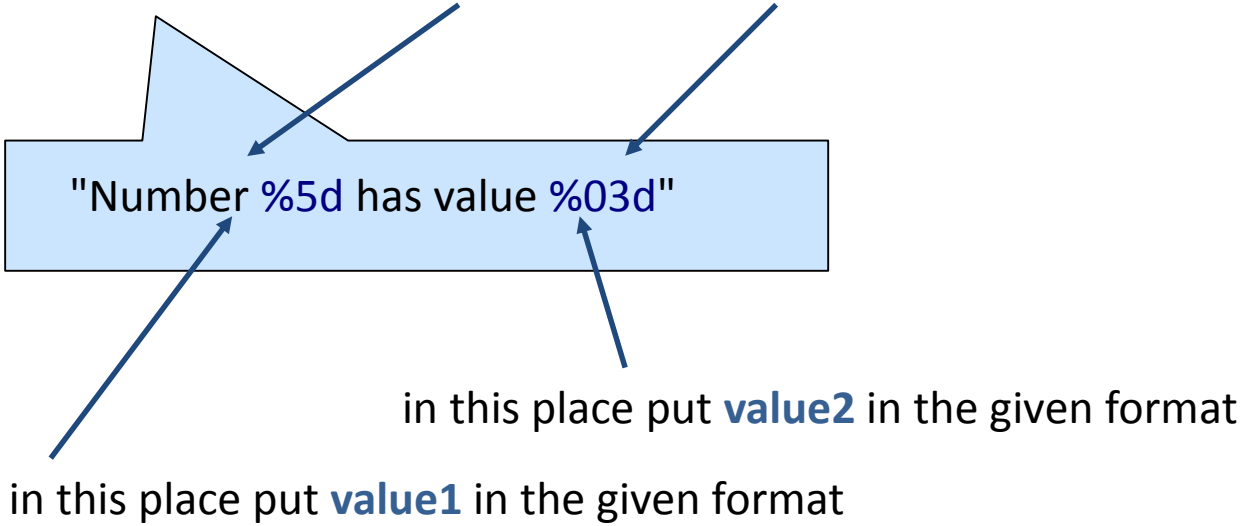
   Verify the result using wc command.
4. Write a script that will calculate the average value of the numbers in the second column of the file matice.txt.

Function printf

Command **printf** serves for formatted output of strings and numbers.

Syntax:

```
printf("format", value1, value2, ...);
```



Comparison with BASH:

```
printf [format] [value1] [value2] ...
```

command

arguments are separated by space

Conditions

```
if(logical_expression) {
      command2;
      ...
} else {
      command3;
      ...
}
```

If **logical_expression** is true, then command **command2** is executed. Otherwise command **command3** is executed.

Example:

```
if( $1 > max ){
      max = $1;
}
```

Comparison with BASH

```
if command1; then
      command2
else
      command3
fi
```

Logical operators

Operators:

==	equal to
!=	not equal to
<	lower than
<=	lower than or equal to
>	greater than
>=	greater than or equal to
!	negation
&&	logical yes
||	logical or

Examples:

```
j > 5
(j > 5)  && (j < 10)
(j <= 5)  || (j >= 10)
```

Loops

```
for(initialization; condition; change) {
      command1;
      ...
}
```

Příklad:

```
for(I=1;I <= 10;I++){
      sum = sum + $I;
}
```

Comparison with BASH

```
for((initialization;condition;change)); do
      command1
done
```

Exercise

1. Write a script which will print the largest and smallest values of the third column in the file matice.txt.
2. Write a script which will print lines from the rst.out file that contain nine word on the line.
3. Write a script which will sum all values in the file matice.txt.