

C2110 UNIX and programming

11th Lesson

Petr Kulhánek, Jakub Štěpán

kulhanek@chemi.muni.cz

National Centre for Biomolecular Research, Faculty of Science
Masaryk University, Kotlářská 2, CZ-61137 Brno



INVESTICE DO ROZVOJE VZDĚLÁVÁNÍ

CZ.1.07/2.2.00/15.0233

Contents

➤ **AWK**

- **What is AWK?**
- **Script structure, script execution**
- **Block structure**
- **Variables, variable operations**
- **Conditions**
- **Cycles**

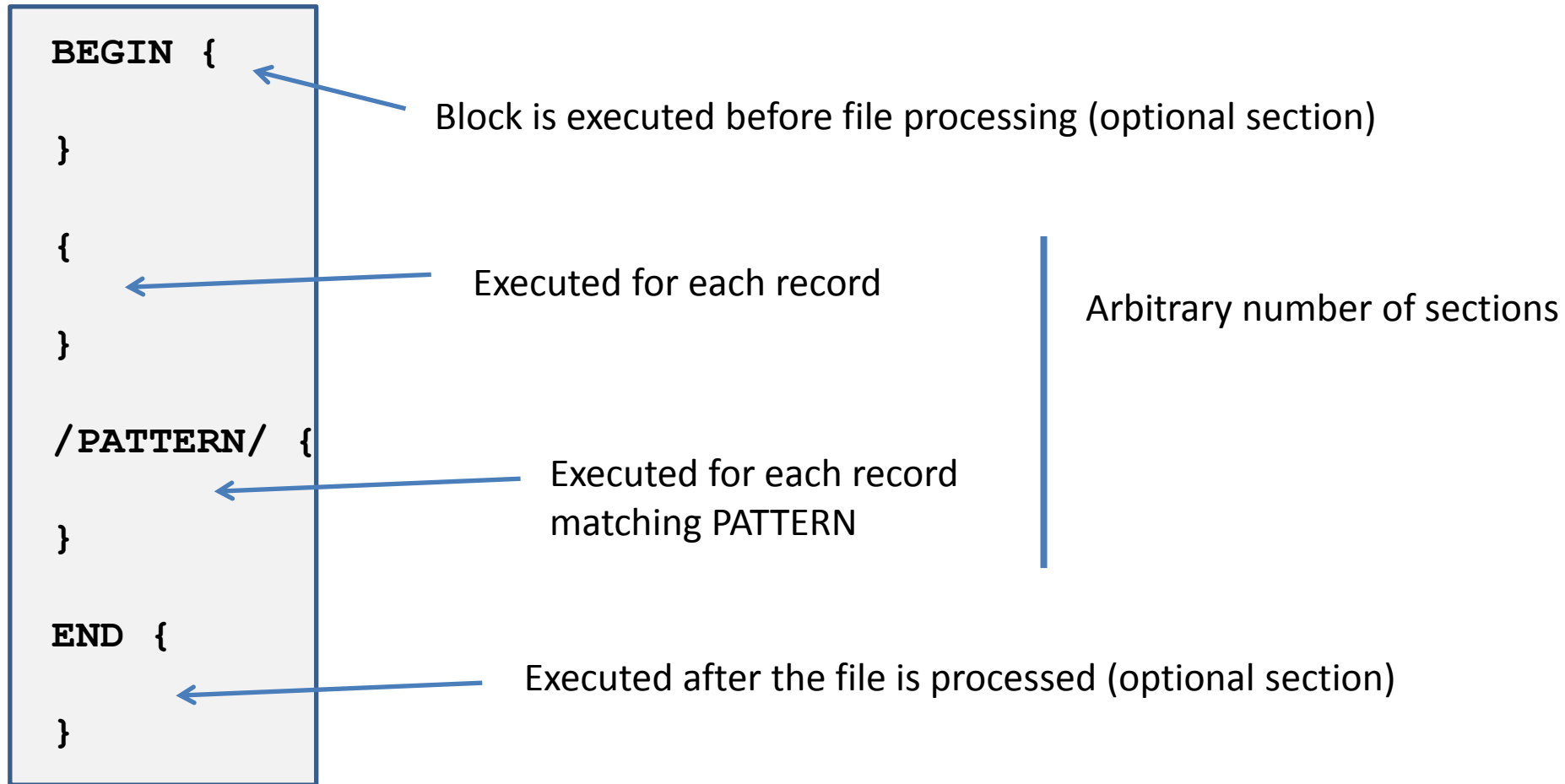
AWK

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

AWK is scripting language designed to **process text data**, either in text files or in streams. Language uses **string data types**, **associative arrays** (arrays indexed by string keys) and **regular expressions**.

Adapted from www.wikipedia.org

AWK script structure

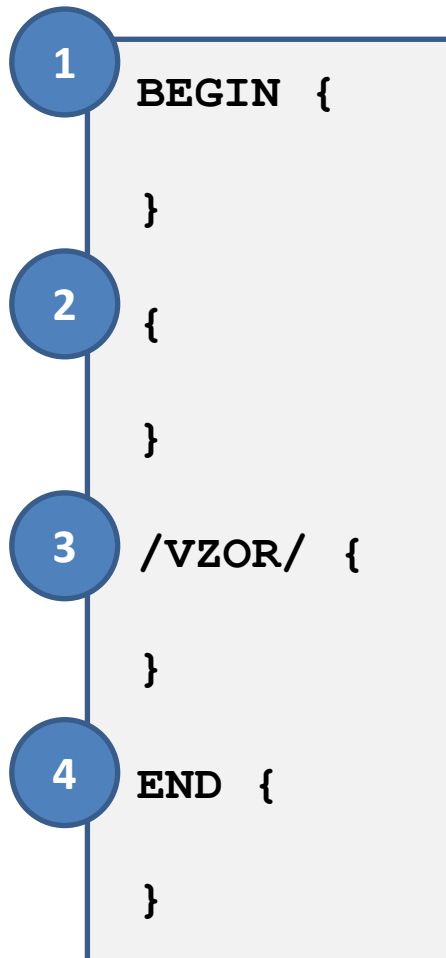


Each block is in curly brackets {}.

Some program blocks are optional – see description.

Default record separator is new line – one line = one record.

Script execution



- Block BEGIN (1) is executed (if present) before file analysis.
 - **Record** from file is read. By default one record is whole line from input file or stream. Record is split to **fields**. By default words of line are fields.
 - Block (2) is executed for **any record**.
 - Block (3) is executed for any **record matching PATTERN**.
 - Possible other blocks are executed
- Block END (4) is executed (if present) after analyzing whole file content.

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 in 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 in 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

Showcase

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;
}
```

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

Comments are denoted by hash symbol #

```
# This block calculates sub-total and
# analyses values in 3rd and 4th column
{
    # comment
    i = i + 1;
    f = f + $2; # sub-total addition
    printf("Sub-total is %10.3f\n", f);
    if( $3 == 5 ) {
        k = k + $4;
    }
}
```

Commands should be on separate lines that may be ended by semicolon.
Semicolon is necessary if multiple commands are on same line.

Variables

Variable assignment:

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

Variable value:

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

Special variables:

NF Number of Fields in current record
NR Number of Record
FS Field Separator, **default is space and tabulator**
RS Record Separator, **default is new line \n**
\$0 Whole current record
\$1, \$2, \$3 ... Particular fields of current record

Must not contain spaces

Difference between BASH

A=5

echo \$A

Variable value accessed by \$

Variables, ...

<code>\$0</code>	Whole record
<code>\$1, \$2, \$3 ...</code>	Particular fields of current record

Symbol `$` enables access to particular record fields in script.

Example:

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



Prints third field value.

Running AWK scripts

Text file processing:

Un-direct running:

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

Language interpreter

awk script

Output is printed to screen

Analyzed text file

Data may be sent through standard input:

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

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

Running AWK scripts, ...

Direct running

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

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

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

script script.awk **needs x (executable)**
permission and interpreter AWK (script first line).

```
#!/usr/bin/awk -f
{
    i += NF;
}
END {
    print "Word count is:", i;
}
```

Exercise

1. Create directory **awk-data** in your home folder.
2. Copy files **matice.txt**, **produkt.log** a **rst.out** from directory **/home/kulhanek/Data/AWK** to directory **awk-data**.
3. Write script, that prints **second column** of file **matice.txt**.
4. Write script, that prints **second and fourth column** from file **matice.txt**.

Math operations

If variable value is in numerical format, following arithmetic operators may be used:

++ Variable value is increased by one

```
A++;
```

-- Variable value is decreased by one

```
A--;
```

+ Sum of two values

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

```
A = A + 1;
```

- Difference of two values

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

```
A = A - 1;
```

***** Multiple of two values

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

```
A = A * 1;
```

/ Quotient of 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** is used for non-formatted print of strings and numbers.

Syntax:

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



If values are separated by comma, in output space is inserted in between them

Examples:

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

Exercise

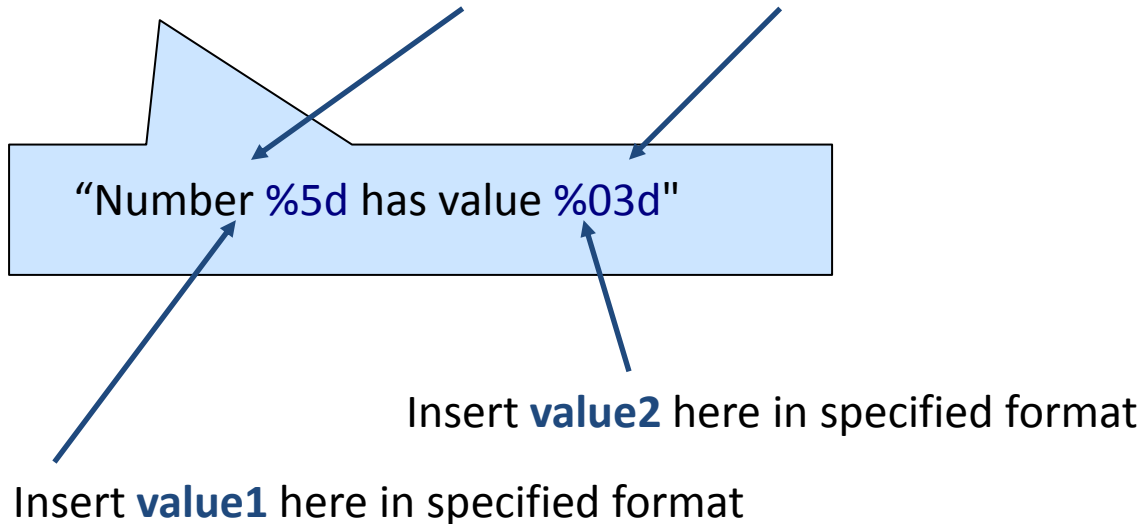
1. Write script, that **calculates sum of numbers in second column** of file **matice.txt**.
2. Write script, that prints **number of lines**, that are in file **matice.txt**. Use command **wc** to verify result.
3. Write script, that print **number of words**, that are in file **matice.txt**. Use command **wc** to verify result.
4. Write script, that calculates **average value** of numbers in second column in file **matice.txt**.

Function printf

Function **printf** prints **formatted** texts and numbers.

Syntax:

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



Difference to BASH:

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

command

Arguments are separated by space

Conditions

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

If **logic_expression** is true, then **command2** is executed. In opposite case **command3** is executed.

Example:

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

Difference to BASH

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

Logic operators

Operators:

<code>==</code>	equal to
<code>!=</code>	not equal to
<code><</code>	less then
<code><=</code>	less or equal
<code>></code>	greater then
<code>>=</code>	greater or equal
<code>!</code>	negation
<code>&&</code>	logical and
<code> </code>	logical or

Examples:

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

Cycles

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

Example:

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

Difference to BASH

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

Exercise

1. Write script, that prints **the greatest and lowest value of third column** in file **matice.txt**.
2. Write script, that prints from file **rst.out particular lines with 9 words**.
3. Write script, that prints **total sum** of all numbers in file **matice.txt**.