

# C2110 *UNIX and programming*

## Lesson 12 / Module 1

**PS / 2020 Distance form of teaching: Rev1**

Petr Kulhanek

[kulhanek@chemi.muni.cz](mailto:kulhanek@chemi.muni.cz)

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

# AWK

---

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

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

adapted from [www.wikipedia.org](http://www.wikipedia.org)

## ➤ AWK

- **Conditions, logical operations**
- **Run control (next, exit)**
- **Loops**
- **Arrays**

# Conditions

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

If **logical\_expression** is true, **command2** will be executed. Otherwise, **command3** will be executed.

Example:

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

## Differences from BASH

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

The diagram shows a code block with three blue arrows pointing to the keywords 'then', 'else', and 'fi' in the code. The text 'Differences from BASH' is written in red above the code block.

# Logical operators

## Operators:

<code>==</code>	equal
<code>!=</code>	not equal
<code>&lt;</code>	smaller than
<code>&lt;=</code>	less than or equal
<code>&gt;</code>	greater than
<code>&gt;=</code>	greater than or equal
<code>!</code>	negation
<code>&amp;&amp;</code>	logical and
<code>  </code>	logical or

## Exampley:

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

# Exercise 1

1. Write a script that prints the largest and smallest value from the third column of the matice.txt file.
2. Write a script that prints lines that contain nine words from rst.out file.
3. Write a script that calculates the average value of the numbers listed in the second column of the matice.txt file.

**The data is in the directory:**

`/home/kulhanek/Documents/C2110/Lesson12`

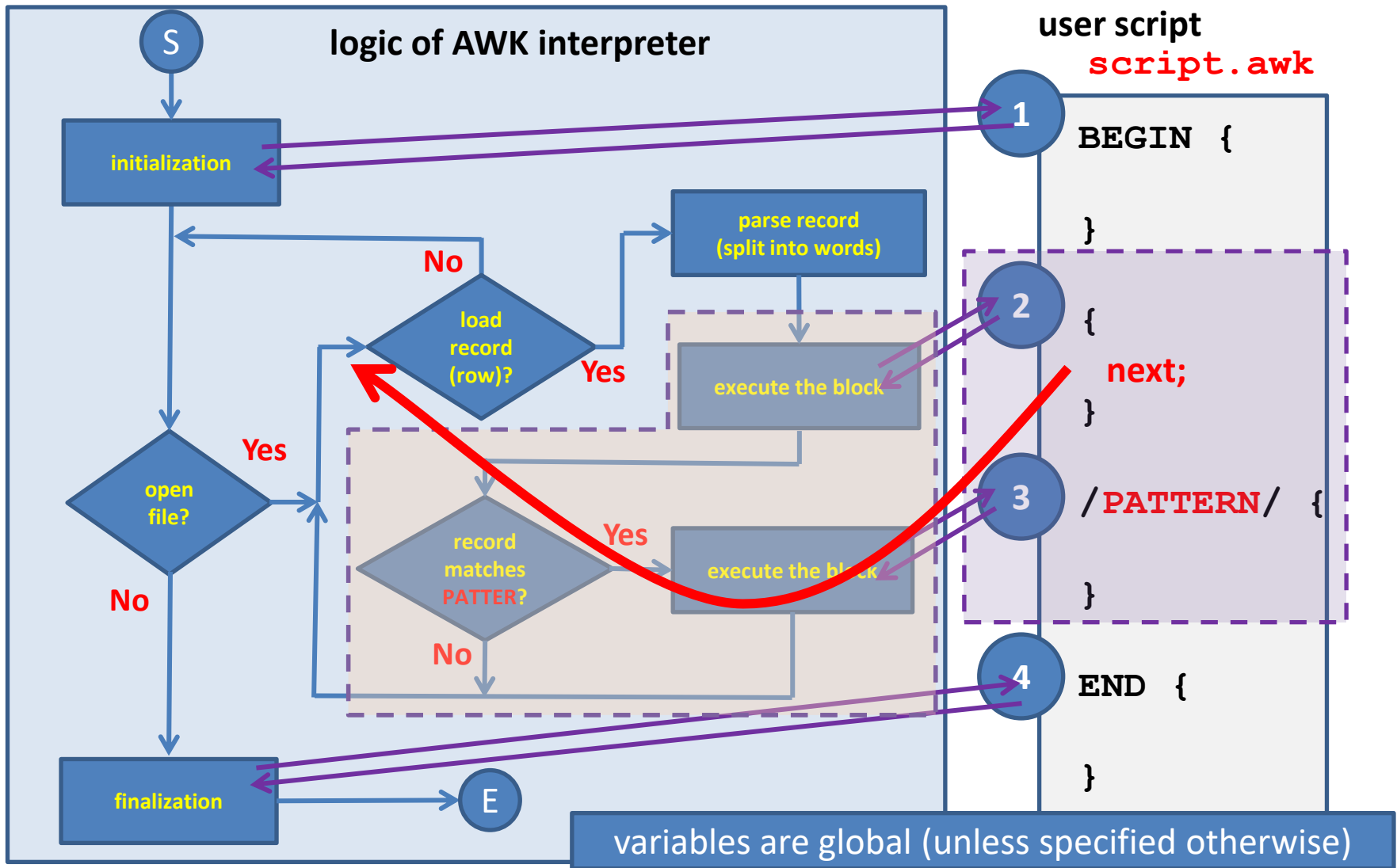
# Exercise 7

1. Write a script that calculates the geometric center of the molecule stored in the format xyz. The resulting coordinates will be printed to the terminal. The file name is entered by the user as the first argument of the script. Take care of situation when the wrong number of arguments is specified and the specified file does not exist. The input file is in a directory geom.

## Help:

- File format xyz contains the number of atoms on the first line , any comment on the second line and the next lines contain the element of the atom and its x, y and z coordinates.
- You can use any combination of commands lines cat, wc, head and tail to discard the first two lines. Alternatively, get inspired on the manual pages of the command tail or apply conditions in awk.

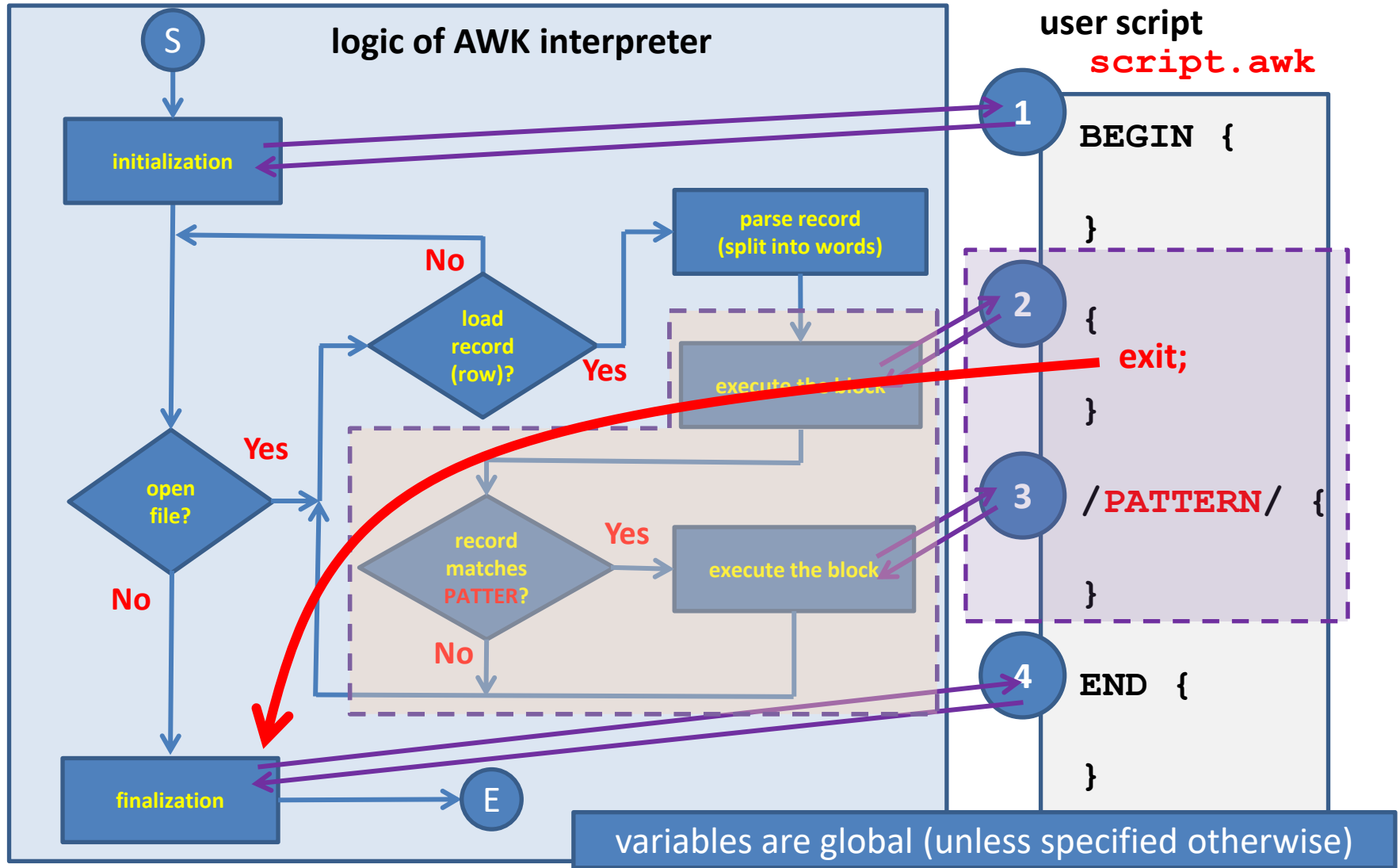
# Running control - next



Key word **next** terminates the processing of the current record. The next entry continues.



# Run control - exit



Key word **exit** stops processing of the current record and all subsequent files.

# Exercise 2

1. Extract temperature profile and calculate its average value from file rst.out. Compare the calculated value with the average value given in the file rst.out. Why do the values differ?

**The data is in the directory:**

`/home/kulhanek/Documents/C2110/Lesson12`

# Loops

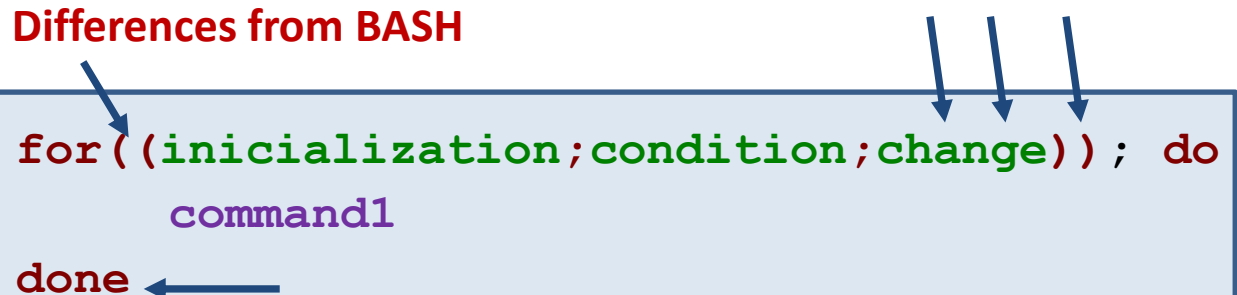
```
for(inicialization; condition; change) {  
    command1;  
    ...  
}
```

Example:

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

Differences from BASH

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

The diagram shows the code from the previous block with four blue arrows pointing to specific parts: one points to the opening parenthesis of the for loop, three point to the semicolons separating the initialization, condition, and change parts, and one points to the 'done' keyword.

# Exercise 3

1. Write a script that sums the values of all the numbers listed in the matice.txt file.
2. Write a script that prints the number of words that the file rst.out contains. Verify the result with the command `wc`.

**The data is in the directory:**

`/home/kulhanek/Documents/C2110/Lesson12`

# Arrays

**AWK** uses associative arrays. The array has a name, the elements of the array are accessed using a key. The key can have any value and type. The key can be the value of a variable.

## Value assignment:

```
my_field[key] = value;
```

## Obtaining value:

```
value = my_field[key];
```

It is not recommended to use  
real numbers as keys!

### Variable: **A**



the variable contains  
**only one** value

```
A= 5;  
print A;
```

### Associative array: **AR**



field may contain **more values**, but only one  
for each key.

```
AR[9] = 5;  
AR["a"] = 10;  
print AR[9], AR["a"];
```

# Arrays - Examples

## Examples:

```
i = 5;
my_array[i] = 15;
print my_array[i];

a = "word";
my_array[a] = "value";
print my_array["word"],          my_array[5];
```

## Practical use:

```
BEGIN {
    count = 0;
}
{
    data[count++] = $1;
}
END {
    print count;
    for(i=0; i < count; i++){
        print data[i];
    }
}
```

script prints number of values in column 1 and then their values

# Exercise 4

1. The structure1.dat file contains the name of the element and the position of the atom on each line. Write a script that converts the file to a format xyz and saves it as structure1.xyz. View the converted structure in VMD.
2. Verify the generality of the solution by converting the structure2.dat file.

```
C -1.8164140 3.6071310 0.6117350
C -1.8002910 2.2769110 0.4584060
C -0.6436270 4.3094580 -0.0124580
.. .. . . .
```

first line: number of atoms

second line: any comment

## Molecule display:

```
$ module add vmd
$ vmd structure1.xyz
```

```
20
molecule
C -1.8164140 3.6071310 0.6117350
C -1.8002910 2.2769110 0.4584060
C -0.6436270 4.3094580 -0.0124580
.. .. . . .
```

**The data is in the directory:**

`/home/kulhanek/Documents/C2110/Lesson12`

# Self-study

---





# Arrays, ...

Browse the key list:

```
for( variable in array) {  
    print array[variable];  
    ...  
}
```

Delete records with key:

```
delete array[key];
```

Executes loop body for each key that was used to store the value in **array**. The key value is stored in **variable**.

**ATTENTION:** order of the keys is not specified and thus may not correspond to order of inserting elements into array