

# C2110 UNIX and programming

## 7<sup>th</sup> Lesson

Petr Kulhánek, Jakub Štěpán

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

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



INVESTMENTS IN EDUCATION DEVELOPMENT

CZ.1.07/2.2.00/15.0233

# Contents

## ➤ Scripts

- Scripting in Bash
- Cycle using for and for in

## ➤ ImageMagic

- Image conversions
- Commands convert, display

## ➤ Commands for names and paths processing

- dirname, basename

# Scripts

---

- **Scripting in Bash**
- **How to write scripts**
- **Cycle using for**


# Script in Bash

```
#!/bin/bash
# this is comment
echo 'This is script in Bash!'
echo "Directory content `pwd` is:"
ls    # print directory contents
A=6   # set variable A value
echo "Variable A value is $A"
echo "first command"; echo "second command"
./mycommand first_argument second_argument \  
    third_argument
```

Command processing order



New line follows



- empty lines are ignored
- text behind symbol # is ignored (used to make comments)
- multiple commands can be on one line, commands are then separated by semicolon ;
- one command may be written on multiple lines using backslash \

# Running Bash scripts

## 1) Indirect run

We run language interpreter and as a parameter we give script name.

```
$ bash my_bash_script
```

Script **does not need** x (executable) permission.

## 2) Direct run

We run directly script (shell runs interpreter automatically).

```
$ chmod u+x my_bash_script
```

```
$ ./my_bash_script
```

Script **needs** permission **x** (executable) and **interpreter** (script file).

```
#!/bin/bash ←
```

```
echo 'This is script in Bash interpreter!'
```

# Cycle using for

Cycle (loop) is control structure, that repeatedly processes set of commands. Repeats are done until condition is fulfilled.

provede se před spuštěním cyklu  
(inicializace počítadla)

If condition is true, commands command1  
etc. are processed.

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

Compact form:

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

Actualization of counting  
variable after commands  
processing

# Cycle using for, example

Write numbers 1 to 10

```
for((I=1;I <= 10;I++)); do
    echo $I
done
```

Variable **I** has **counting** role.

**Initialization** variable assignment is same as Bash variable assignment.

## Change:

If variable may be interpreted as number, then following arithmetic operators may be used:

- ++** increments variable value by one
- decrements variable value by one

To be continued ....

Write numbers 10 to 1

```
for((I=10;I >= 1;I--)); do
    echo $I
done
```

## Condition:

If variable may be interpreted as number, following operators can be used:

<b>!=</b>	not equal
<b>==</b>	equal
<b>&lt;</b>	lower
<b>&lt;=</b>	lower or equal
<b>&gt;</b>	greater
<b>&gt;=</b>	greater or equal

# Cycle using for, count change

If variable may be interpreted as number, then following arithmetic operators may be used:

**++** increments variable value by one

**A++**

**--** decrements variable value by one

**A--**

**+** adds 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 (integer division)

**A=5 / 6**

**A=A / 1**

**A=A+3**

**+=** 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**





# Cycle and variables

## Correct syntaxes

### Count initialization:

```
A=1
```

```
A = 1
```

### Condition:

```
A <= 10
```

```
$A <= 10
```

### Count change:

```
A=$A + 1
```

```
A = $A + 1
```

```
A=A + 1
```

```
A = A + 1
```

It is possible to use space between symbol =, variable name and value.

It is optional to use operator \$ for access to variable value (not needed).

Benevolent syntax is only in cycle definition.

```
for((I=1;I <= 10;I++)); do  
    echo $I  
done
```

Here operator is compulsory to access variable value.

# Exercise

1. Write script, that writes path to current directory and its contexts.
2. Write ten symbols **A**, each to separate line.
3. Write ten symbols **A** all on same line (**echo -n** and manual page).
4. Write script, that prints even numbers from **2** to **100**.
5. Write script, that prints power  $2^n$  for **n** from **0** to **32**.

## Reccomendation:

Solve each task in separate directory. Number your directories (for example):

task01

task02

etc.

# Nested cycle

Cycle control structures may be nested.

```
for((I=1;I <= 10;I++)); do
  for((J=1;J <= 10;J++)); do
    echo "$I $J"
  done
done
```

Outer cycle

Inner cycle

```
for((I=1;I <= 10;I++)); do
  for((J=1;J <= I;J++)); do
    echo "$I $J"
  done
done
```

Count of outer cycle may influence behavior of inner cycle.

# Exercise

1. Compare function of nested cycles shown on previous page.
2. Write script, that print ten times ten symbols (of your choice) on line.
3. Write script, that print one symbol on first line, two symbols on second line, etc. Write ten lines in this way.
4. Change previous script, so that it print 15 times 6 symbols on one line.
5. Find out why it is advantageous to use variables in control structures.

Values in control structures should be given by variables.

```
for((I=1;I <= 10;I++)); do  
    echo $I  
done
```

**Worse solution**

```
NUMBER=10  
for((I=1;I <= NUMBER;I++));  
do  
    echo $I  
done
```

# Image Magic

---

## ➤ Image conversion

<http://www.imagemagick.org>

(documentation, tutorials, source code)



# Commands

## Overview:

animate, compare, composite, conjure, **convert**, **display**, identify, import, mogrify, montage, stream

Detail description is accessible in manual pages or web pages of Image Magic.

## Most important commands:

**display** show image or image sequence on screen

**convert** converts image between formats, including operations like size change, crop, etc.

## Examples:

```
$ convert input.eps output.png
    convert image from postscript format to PNG format
```

## High quality for publications:

```
$ convert -density 300x300 input.eps -units PixelsPerInch \  
    -density 300 -background white -flatten output.png
```

# Scripts

---

- Cycle using for in

# Cycle using for ... in ...

Commands in block **do/done** (**command1**, ...) are processed for each item in list **LIST**. In each loop iteration variable **VAR** contains actual item from list **LIST**.

```
for VAR in LIST
do
    command1 $VAR
    ...
done
```

Compact syntax:

```
for VAR in LIST; do
    command1 $VAR
    ...
done
```



# Cycle using for ... in ..., lists

```
for A in a b c; do
    echo $A
done
```

Cycle is done three times, printing symbols **a**, **b**, **c** consequently.

Lists of items may be created by program (using backward apostrophes).

```
for A in `ls *.eps`; do
    ./process_file $A
done
```

Command **process\_file** is processed for each file with extension **.eps** that is in current working directory.

# Commands for paths and names processing

---

- **dirname**
- **basename**

# Commands for name and path proc

**dirname**                extracts directory name from full path  
**basename**             extracts file name from full path

## Example:

```
$ basename /home/kulhanek/pokus.txt  
pokus.txt  
  
$ basename pokus.txt  
pokus.txt  
  
$ basename /home/kulhanek/pokus.txt .txt  
pokus  
  
$ dirname /home/kulhanek/pokus.txt  
/home/kulhanek  
  
$ dirname pokus.txt  
.
```

Commands **dirname** and **basename** process paths without checking if file exists.

# Exercise

1. Create directory **images**
2. Copy files with extension **.eps** from **/home/kulhanek/Data/Snapshots/** to directory **images**.
3. Write script, that prints file names, that contain directory **images** in following format:  

```
Directory images contain file: file1.eps  
Directory images contain file: file2.eps
```
4. Write script that, convert files from format **eps** in directory **images** to format **png**.
5. Make sure by command **display**, that conversion was processed correctly.