

C2110 *UNIX and programming*

Lesson 8 / Module 2

PS / 2020 Distance form of teaching: Rev2

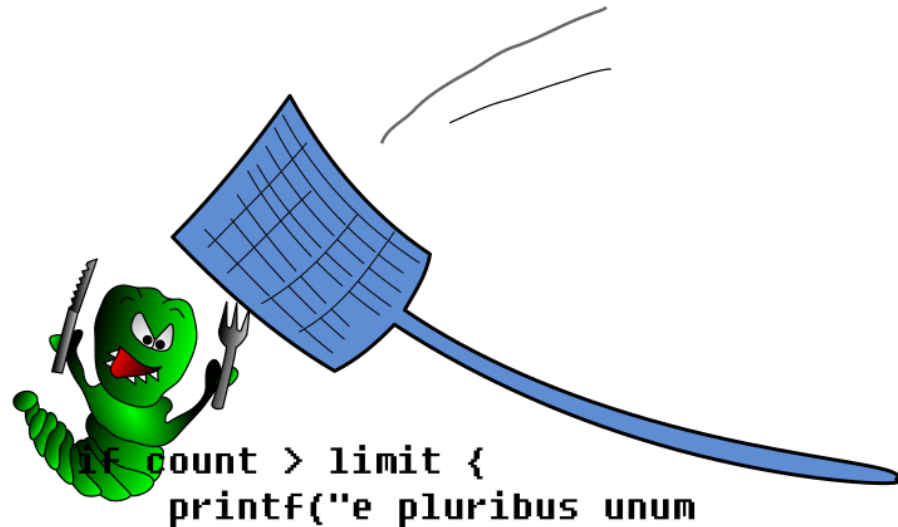
Petr Kulhanek

kulhanek@chemi.muni.cz

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

Troubleshooting

- Syntax errors
- Logical errors



Syntax Errors

Syntax errors

- writing errors that do not match the language specification
- program cannot be translated, place of error is printed by compiler (e.g., C/C++, Fortran)
- script cannot be run at all (e.g. ,javascript)
- when interpreter encounters an incomprehensible notation during the run of the script, the run is terminated and the line with the error is written to the error output, the error may manifest only under specific run conditions (e.g., bash, awk, gnuplot)

Attention!

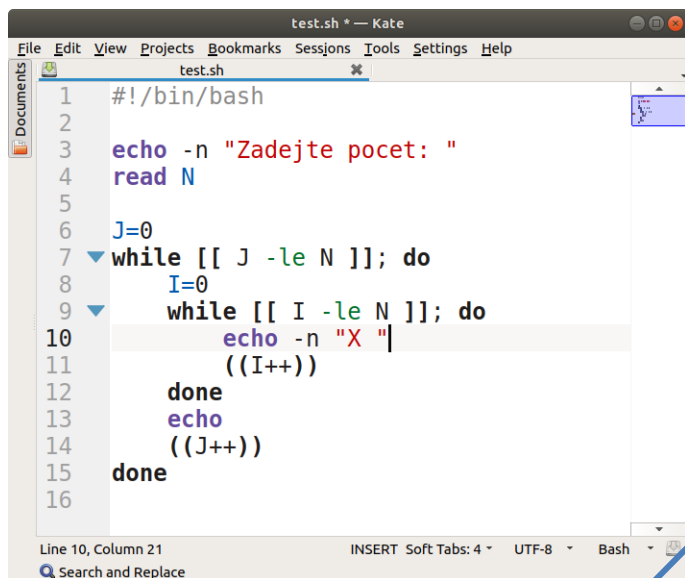
Beginning of the error may be at a different location than indicated by the error message.

Troubleshooting:

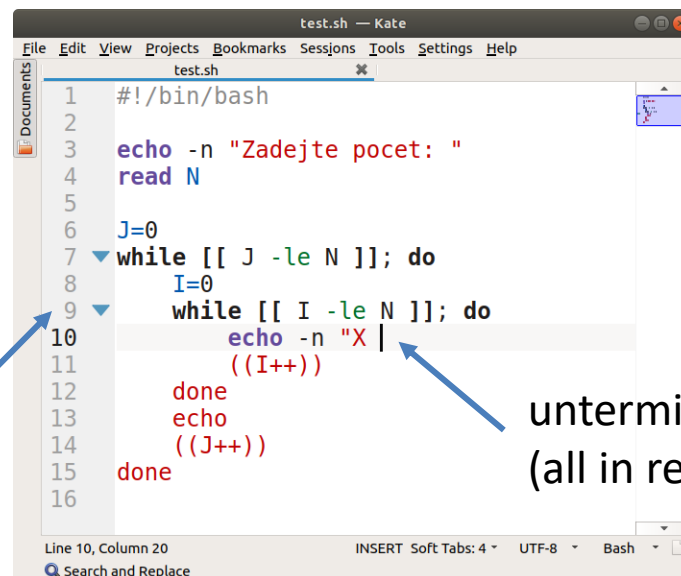
- syntax highlighting
- static code analyzers

bash - Syntax Check

1. The basic syntax check can be performed visually **in a text editor**, which allows **highlighting syntax** of programming language.



```
1 #!/bin/bash
2
3 echo -n "Zadejte pocet: "
4 read N
5
6 J=0
7 while [[ J -le N ]]; do
8     I=0
9     while [[ I -le N ]]; do
10         echo -n "X |"
11         ((I++))
12     done
13     echo
14     ((J++))
15 done
16
```



```
1 #!/bin/bash
2
3 echo -n "Zadejte pocet: "
4 read N
5
6 J=0
7 while [[ J -le N ]]; do
8     I=0
9     while [[ I -le N ]]; do
10         echo -n "X |"
11         ((I++))
12     done
13     echo
14     ((J++))
15 done
16
```

unterminated string
(all in red)

2. **Error message:**

turn on display of line numbers

```
$ bash test.sh
```

```
Zadejte pocet: 5
```

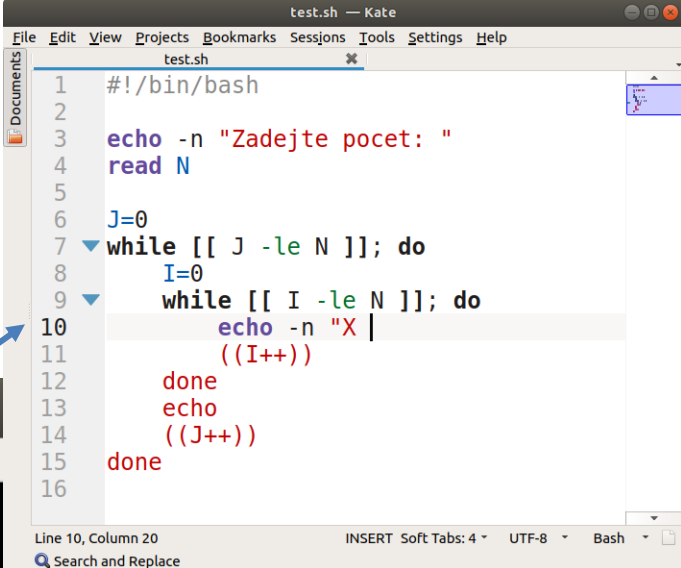
```
test.sh: line 10: unexpected EOF while looking for matching `''
```

```
test.sh: line 16: syntax error: unexpected end of file
```

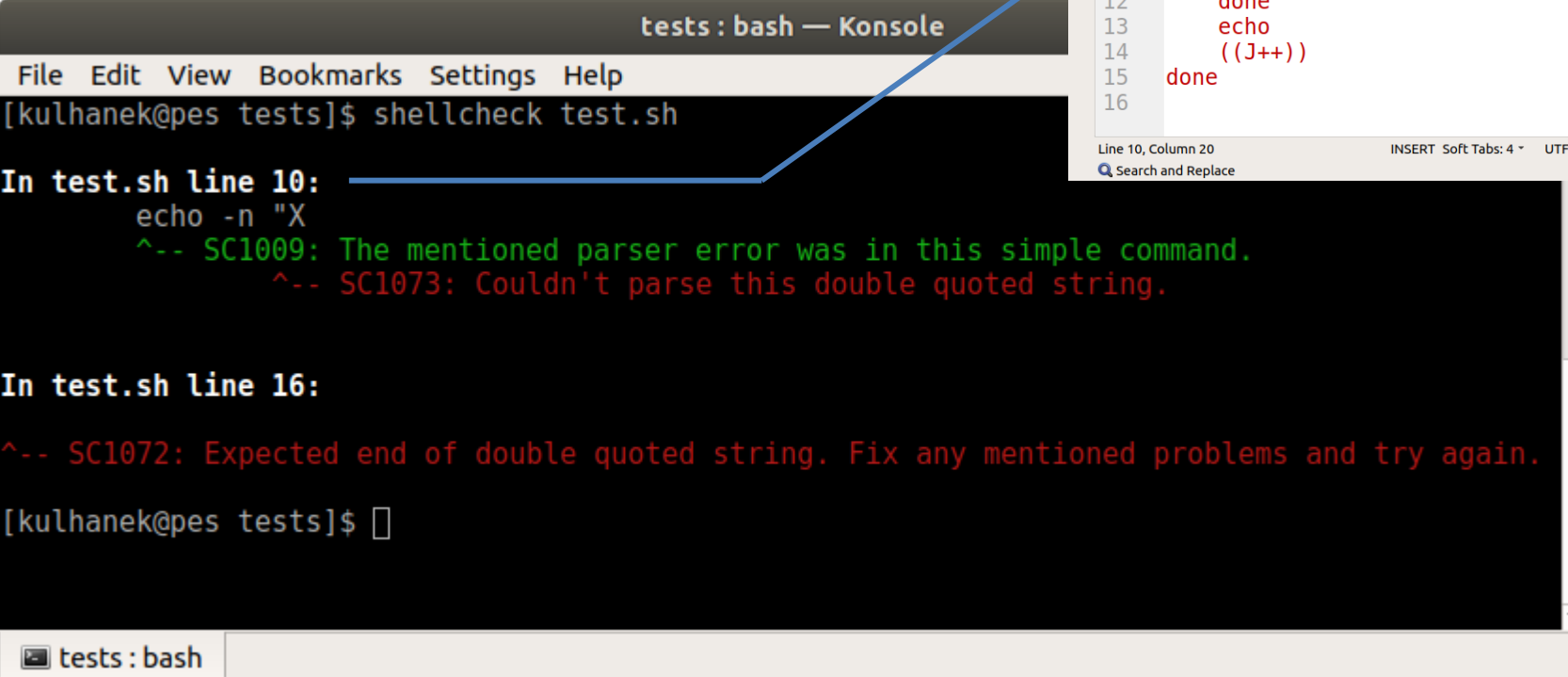
bash - Static Analysis

3. Static code analyzer:

- online
<https://www.shellcheck.net/>
- command line



```
test.sh — Kate
File Edit View Projects Bookmarks Sessions Tools Settings Help
test.sh
1 #!/bin/bash
2
3 echo -n "Zadejte pocet: "
4 read N
5
6 J=0
7 while [[ J -le N ]]; do
8     I=0
9     while [[ I -le N ]]; do
10        echo -n "X |
11            ((I++))
12        done
13        echo
14        ((J++))
15    done
16
```



```
tests : bash — Konsole
File Edit View Bookmarks Settings Help
[kulhanek@pes tests]$ shellcheck test.sh

In test.sh line 10:
  echo -n "X
  ^-- SC1009: The mentioned parser error was in this simple command.
     ^-- SC1073: Couldn't parse this double quoted string.

In test.sh line 16:
^-- SC1072: Expected end of double quoted string. Fix any mentioned problems and try again.

[kulhanek@pes tests]$
```

Logical Errors

Logical errors

- program/script can be run, but the result does not meet expectations or is not reproducible
- these errors are **VERY hard to find/fix** - it is advisable to avoid them by thorough design of the algorithm

Causes of logical errors:

- bad algorithm design (all languages)
- work with uninitialized variables (all languages)
- work with unallocated/freed memory, writing to unallocated memory (C/C++, Fortran)
- concurrence (race condition) for parallel tasks (OpenMP, MPI, threads)

Troubleshooting:

- dynamic run analyzers (C/C++, Fortran: valgrind)
- debugging the program run using a debugger

debugger - bashdb + visual code

1. Starting the editor:

```
$ module add vscode  
$ code
```

2. Install the extension (only once):

<https://marketplace.visualstudio.com/items?itemName=rogalmic.bash-debug>

3. Runtime environment configuration (launch.json, only once):

```
{  
  // Use IntelliSense to learn about possible attributes.  
  // Hover to view descriptions of existing attributes.  
  // For more information, visit: https://go.microsoft.com/fwlink/?linkid=830387  
  "version": "0.2.0",  
  "configurations": [  
    {  
      "type": "bashdb",  
      "request": "launch",  
      "name": "Bash-Debug",  
      "cwd": "${workspaceFolder}",  
      "program": "${file}",  
      "args": [],  
      "terminalKind": "integrated"  
    },  
  ],  
}
```

Practical example of configuration and use.

debugger - bashdb + visual code

Debugger

Values of variables
(operator: \$)

The screenshot shows the Visual Studio Code interface with the following components:

- Debugger:** Located in the left sidebar, it shows the 'VARIABLES' section with local variables: `$PWD: /home/kulhanek/DownL...` and `$?: 0 # from '[' I -le N ...`. Below it is the 'WATCH' section showing variable values: `$J: '0'`, `$I: '5'`, and `$N: '5'`.
- Code Editor:** Displays a bash script named `test.sh`. The current execution position is highlighted in yellow on line 10: `echo -n "X "`. The script includes a `while` loop with nested `while` loops and `echo` statements.
- Terminal:** Shows the output of the script: `bash -c "cd \"/home/kulhanek/Downloads/bashdb/tests\"; while [[! -p \"/tmp/vsc...` followed by `Zadejte pocet: 5` and `X X X X X`.
- Annotations:** Blue arrows point to the 'Debugger' sidebar, the 'WATCH' section, the 'Current position' (line 10), the 'Terminal: input/output' area, and the 'Step Over' button in the top right of the editor.

stops
(breakpoints)

Current position

Step Over

Terminal: input/output

Exercise I

1. Step through the running scripts from L08.M01.E01.T03.

Installation

Installation of shellcheck and bashdb to the Ubuntu OS



Notes to Installation

1. **shellcheck** is part of standard packages:

```
$ sudo apt-get install shellcheck
```

2. **bashdb** can be installed from source code (see L13.M02) or into Ubuntu 18.04 LTS from NCBR package repository, package name: ncb-r-bashdb

<https://wolf.ncbr.muni.cz/whitezone/packages/public/18.04/>

3. **Visual Studio Code** can be installed from package (.deb) or run from a binary archive (.tar.gz). See the documentation for instructions.

<https://code.visualstudio.com/>

For Infinity users (C2115): bashdb and extension bash-debug require a system command pkill which is redefined in the Infinity environment.

Resolve the collision by running unset command before starting the editor.

```
$ unset pkill
```