C2110 UNIX and programming

5th Lesson

Petr Kulhánek, Jakub Štěpán

kulhanek@chemi.muni.cz

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

european social fund in the czech republic
EUROPEAN UNION
MINISTRY OF EDUCATION, YOUTH AND SPORTS
OP Education for Competitivness
INVESTMENTS IN EDUCATION DEVELOPMENT

CZ.1.07/2.2.00/15.0233

Contents

In-semester test I

In-semester test I

- **Test is questionaire (ROPOT) in IS**

  Student – ROPOT – e-learning – C2110 – In-semester test I

  Length 20 minutes.

  Only one set of questions.

  Use 'Save temporarily' during work.

  Evaluation can be done only once.

  It is allowed and suggested to

  Test commands in terminal.
  Search manual pages, lecture notes and lecture presentations.
  Call teacher if you have problems.

  It is forbidden to

  Communicate with other person except teacher

Text editors

- vi, vim, nano
- Graphical text exitors
  - kwrite, gedit, kate

vi/vim, nano

Editor vi / vim is standard in operating systems of UNIX type. Only in text mode and usage is **non-trivial.**

- It is useful to learn to open file, edit text, save changes and close editor.
- Enables scripting (using variables, cycles, arrays, associative arrays). Can be used for example to automatic generation of text from data read.
- Although you run command vi on WOLF cluster, program **vim** (Vi IMporoved)
- There are control differences between **vi** and **vim**.

Editor nano is default text editor on some distributions.

- Not so universal and flexible as vim.
- Straightforward control.

vi – basics

Editor work modes

$$\begin{tikzcd}
\text{Extended command mode} \arrow[r, shift right, "\text{Enter}"'] & \text{Command mode} \arrow[l, shift right, "\text{:}"'] \arrow[r, shift left, "\text{a, c, i, o, s, A, C, I, O, R, S}"] & \text{Input mode} \arrow[l, shift left, "\text{Esc}"]
\end{tikzcd}$$

Running editor

$ vi	**start** editor
$ vi *filename*	**start** editor and **open file** filename

Closing editor

:q	**close** editor
:q!	**close editor without saving changes**
:w	**save file**
:w *filename*	**save** data to file *filename*
:wq	**close editor saving changes**

Changes in file

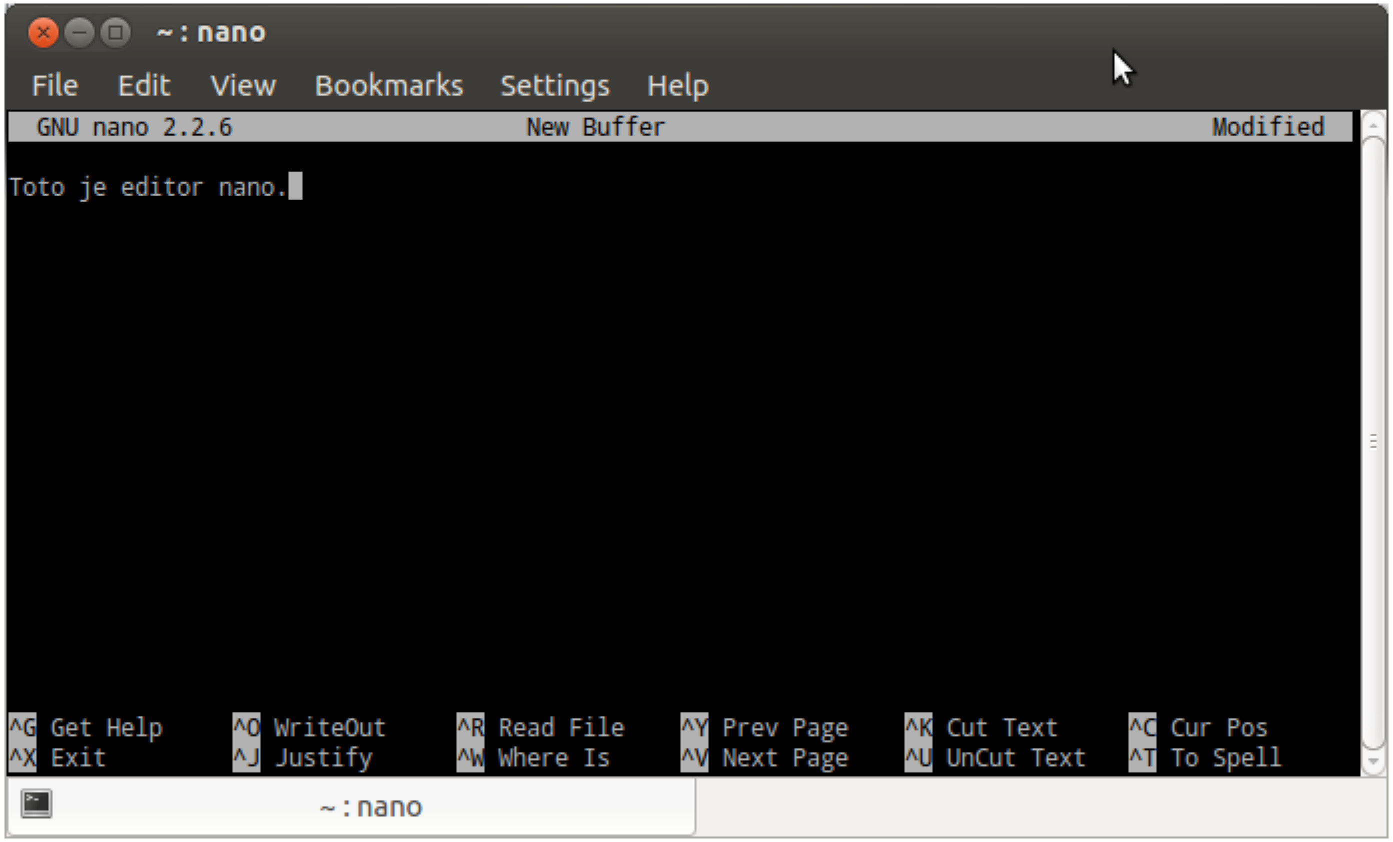
i	text will be places **from** cursor position
a	text will be places **after** cursor position

nano

Straightforward control – menu in bottom part helps with control

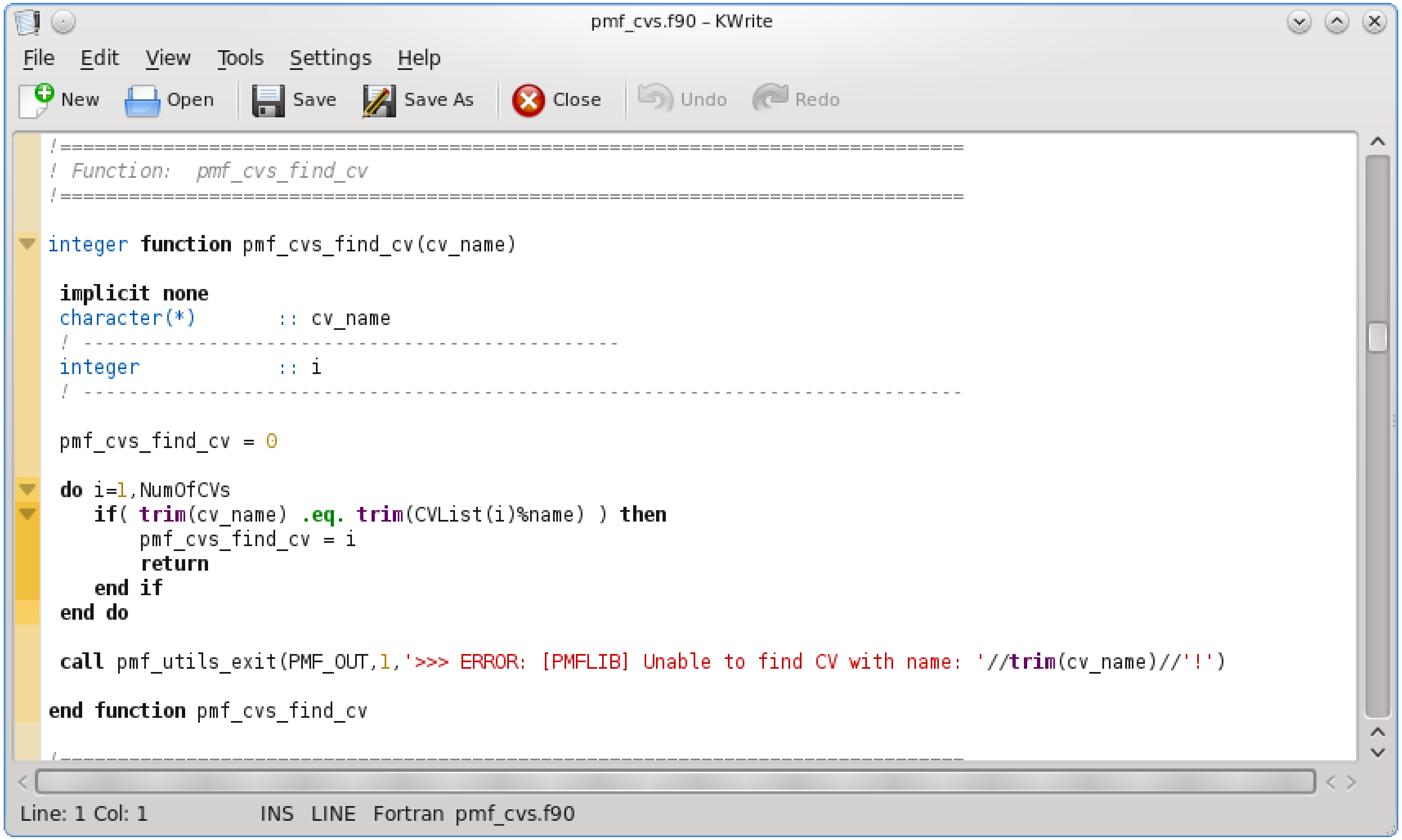
Action is called by single keys or key combinations

- **^character – e.g. ^X** means combination **Ctrl + X**
- **M-character – e.g. M-M** means combination **Alt+M**



```
  GNU nano 2.2.6                New Buffer                          Modified

Toto je editor nano.

^G Get Help    ^O WriteOut    ^R Read File   ^Y Prev Page   ^K Cut Text    ^C Cur Pos
^X Exit        ^J Justify     ^W Where Is    ^V Next Page   ^U UnCut Text  ^T To Spell
```

kwrite

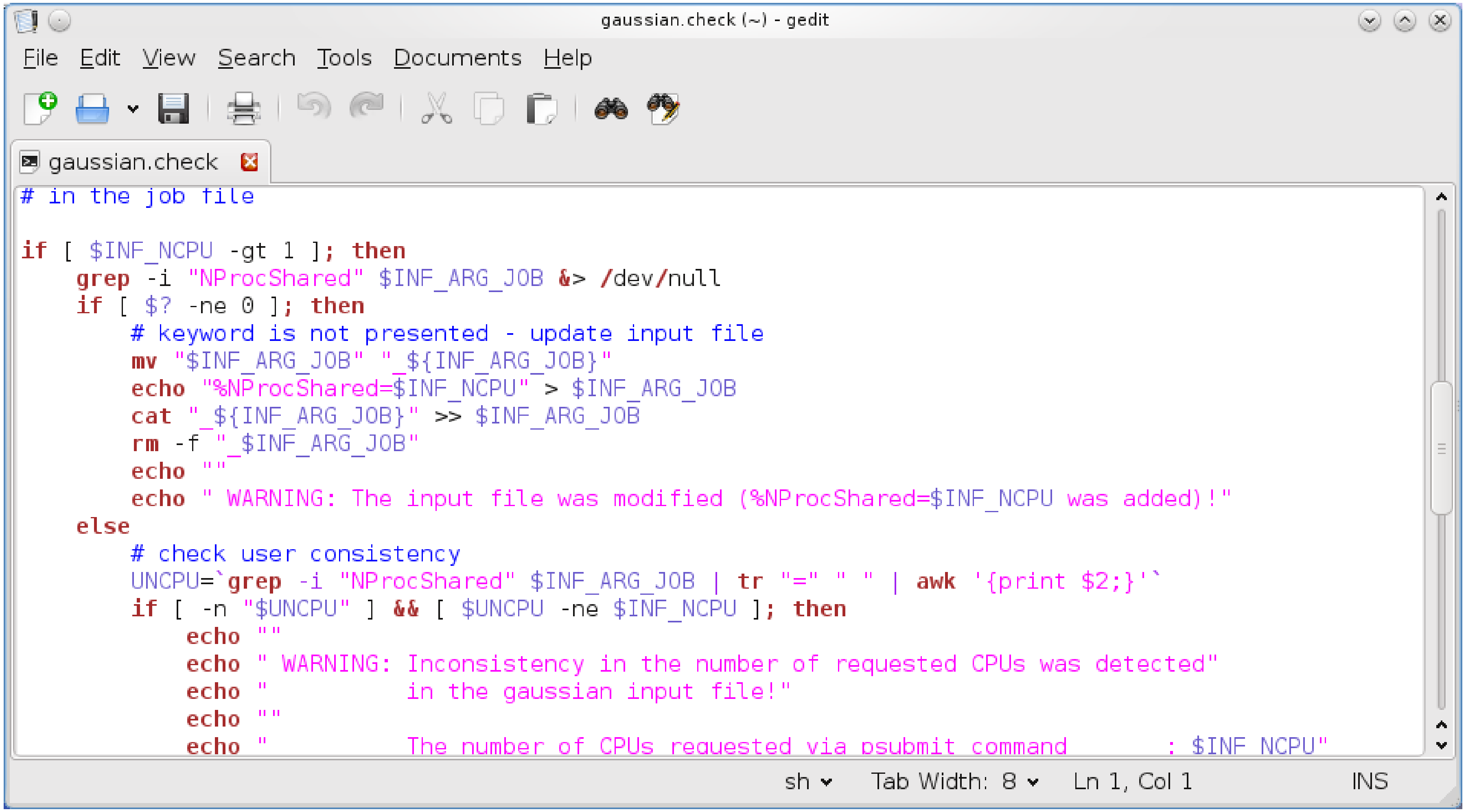

pmf_cvs.f90 – KWrite
File Edit View Tools Settings Help
New Open Save Save As Close Undo Redo

```
!==============================================================================
! Function:   pmf_cvs_find_cv
!==============================================================================

integer function pmf_cvs_find_cv(cv_name)

 implicit none
 character(*)          :: cv_name
 ! -----------------------------------------------
 integer               :: i
 ! ------------------------------------------------------------------------------

 pmf_cvs_find_cv = 0

 do i=1,NumOfCVs
    if( trim(cv_name) .eq. trim(CVList(i)%name) ) then
        pmf_cvs_find_cv = i
        return
    end if
 end do

 call pmf_utils_exit(PMF_OUT,1,'>>> ERROR: [PMFLIB] Unable to find CV with name: '//trim(cv_name)//'!')

end function pmf_cvs_find_cv
```

Line: 1 Col: 1   INS   LINE   Fortran pmf_cvs.f90

Extended version: **kate**

gedit


gaussian.check (~) - gedit
File Edit View Search Tools Documents Help
gaussian.check
sh
Tab Width: 8
Ln 1, Col 1
INS

```
# in the job file

if [ $INF_NCPU -gt 1 ]; then
    grep -i "NProcShared" $INF_ARG_JOB &> /dev/null
    if [ $? -ne 0 ]; then
        # keyword is not presented - update input file
        mv "$INF_ARG_JOB" "_${INF_ARG_JOB}"
        echo "%NProcShared=$INF_NCPU" > $INF_ARG_JOB
        cat "_${INF_ARG_JOB}" >> $INF_ARG_JOB
        rm -f "_$INF_ARG_JOB"
        echo ""
        echo " WARNING: The input file was modified (%NProcShared=$INF_NCPU was added)!"
    else
        # check user consistency
        UNCPU=`grep -i "NProcShared" $INF_ARG_JOB | tr "=" " " | awk '{print $2;}'`
        if [ -n "$UNCPU" ] && [ $UNCPU -ne $INF_NCPU ]; then
            echo ""
            echo " WARNING: Inconsistency in the number of requested CPUs was detected"
            echo "          in the gaussian input file!"
            echo ""
            echo "          The number of CPUs requested via psubmit command       : $INF_NCPU"
```

Exercise

1. Write text with **ten lines** in **vi** editor. Each line will have **two or more** words. Save text into file **mydata.txt**
2. Use command **wc** to make sure that **mydata.txt** has exactly **ten** lines.
3. Use pipe(s) to construct command sequence, to print only number of words in file **mydata.txt**
4. Create text file in graphics text editor (your choice) containing **ten** words, each word on **separate** line. Save text to file **second_data.txt**
5. Use command **paste** to create file **all_data.txt** , that contains data from files **mydata.txt** and **second_data.txt next to each** other.
6. Use command **wc** to make sure, that file **all_data.txt** contains **ten** lines.
7. Open file **all_data.txt** in graphical text editor and **check contents**.
8. Open file **all_data.txt** in editor **nano** and save to new file name in **mac** format, what is **difference** to original file? Print contents of both files by **cat**, open in **vi** or **gedit**.

Processes II

- Commands
- Running commands and applications
- Killing commands and applications

Commands

top	prints processes by CPU time consumption – periodic refresh (finish by key q)
ps	print processes running in terminal (options can print all processes and various information) (ps -u user_name)
pstree	process tree print
kill	sends signal to process (default signal is TERM), used to terminate problematic processes
nohup	runs process without terminal interaction
sleep	runs process, that waits for specified time
wait	wait for background processes to finish
time	writes process run time
ssh	run process on remote machine, login to remote machine
jobs	prints list of background processes
fg	switches process from background to foreground
bg	switches process from foreground to background
disown	detach process from terminal

Running commands & applications

System commands and applications

```
$ ls -l
$ cp file.txt file1.txt
```

příkaz

Call by command or application name

Command options parameters (change command behavior and are input data of command processing)

User program and scripts

```
$ ./my_script
$ ~/bin/my_application
```

Program or script name has to be with **full path** (absolute or relative)

Redirect (discard) standard output to terminal

```
$ kwrite &> /dev/null
```

Output redirection is given on the end of command line (after parameters)

Run command on background

```
$ gimp &
```

Ampersand - & on the end runs command on background (after parameters and redirections)

Running commands & applications II

Terminal (useful key shortcuts):

Ctrl+C	sends signal SIGINT (Interrupt) to running process, process is usually terminated immediately
Ctrl+D	close input stream of running process
Ctrl+Z	pause process run, following process management can be done by commands **bg**, **fg**, **disown**

Print full path to system command:

type	print path to system command or program

Examples:

```
$ type ls
ls is /bin/ls

$ type pwd
pwd is a shell builtin
```

Command is implemented as inner shell command (builtin)

Examples

```
$ ps -u kulhanek
 PID TTY          TIME CMD
...
 5440 pts/8    00:00:00 bash
 5562 pts/8    00:00:00 kwrite
 5566 pts/8    00:00:00 ps

$ kill 5562   # terminate kwrite application

$ kwrite      # run kwrite application on foreground
^Z            # pause application run
[1]+  Stopped                 kwrite
$ jobs        # print list of applications on background
[1]+  Stopped                 kwrite
$ bg 1        # application 1(kwrite) is switched to foreground
[1]+ kwrite &
$ jobs
[1]+  Running                 kwrite &
```

Exercise

1. Measure time length of **sleep 0,003** process run, how long is it, why?
2. Get name of process number **1**, who is process owner?
3. Try to kill the process, **why** is it not possible?