C2110 *Operační systém UNIX a základy programování*

5. lekce

Petr Kulhánek, Jakub Štěpán

kulhanek@chemi.muni.cz

Národní centrum pro výzkum biomolekul, Přírodovědecká fakulta
Masarykova univerzita, Kotlářská 2, CZ-61137 Brno

INVESTICE DO ROZVOJE VZDĚLÁVÁNÍ

CZ.1.07/2.2.00/15.0233

Obsah

- Průběžný test
- Textové editory
  - vi, vim, nano
  - grafické textové editory (kwrite, gedit, kate)
- Procesy II
  - příkazy
  - spouštění příkazů a aplikací
  - ukončovaní příkazů a aplikací

Průběžný test I

Průběžný test I

➢ Test prostřednictvím odpovědníku v IS

Student – Odpovědníky – C2110 – Průběžný test I

Délka 20 minut.

Je možné sestavit pouze jednu sadu otázek.

Používejte průběžné uložení.

Vyhodnocení je možné pouze jednou.

Je povoleno a doporučeno:

- Testovat příkazy v terminálu.
- Prohledávat manuálové stránky, svoje zápisky a prezentace předmětu.
- Při nejasnostech se přihlaste.

Není povoleno

- Komunikovat s další osobou mimo vyučujícího.

Textové editory

- vi, vim, nano
- grafické textové editory (kwrite, gedit, kate)

vi/vim, nano

Editor vi / vim je standardním textovým editorem v operačních systémech UNIXového typu. Pracuje pouze v textovém módu a jeho používání je **netriviální**.

- Je vhodné se naučit, jak otevřít soubor, přejít do editačního módu, upravovat text, uložit provedené změny a editor ukončit.
- Umožňuje skriptování (použití proměnných, cyklů, polí, asociativních polí) např. pro vytvoření automatických textů z načtených dat.
- Přestože v učebně budete spouštět příkaz vi, automaticky se spustí program vim (Vi IMporoved)
- Mezi původním vi a vim je rozdíl v ovládání.

Editor nano je výchozím textovým editor v některých distribucích (UBUNTU).

- Méně univerzální než vim
- Přímočařejší ovládání

vi – základy

Pracovní módy editoru

Spuštění editoru

$ vi	**start** editoru
$ vi *filename*	**start** editoru a **otevření souboru** filename

Ukončení editoru

:q	**ukončení** editoru
:q!	**ukončení editoru bez uložení** změn
:w	**uložení** souboru
:w *filename*	**uložení** souboru po jménem *filename*
:wq	**ukončení s uložením** souboru

Změny souboru

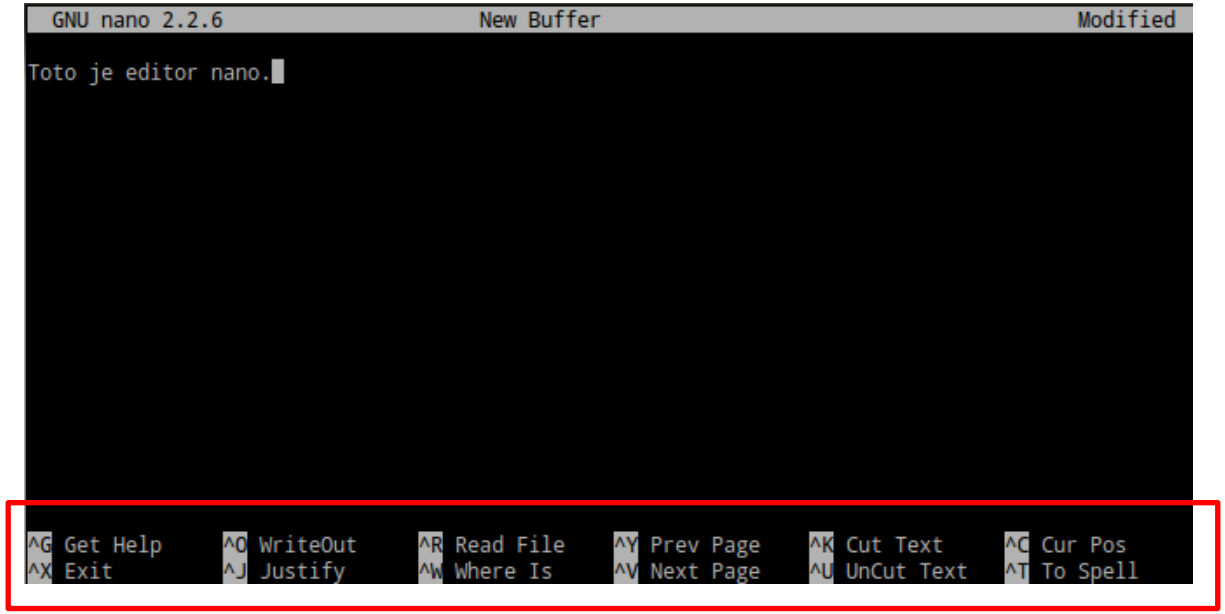
i	text bude vkládán **od** pozice kurzoru
a	text bude vkládán **za** pozici kurzoru

nano

Spuštění editoru

$ nano **start** editoru

$ nano *filename* **start** editoru a **otevření souboru** filename

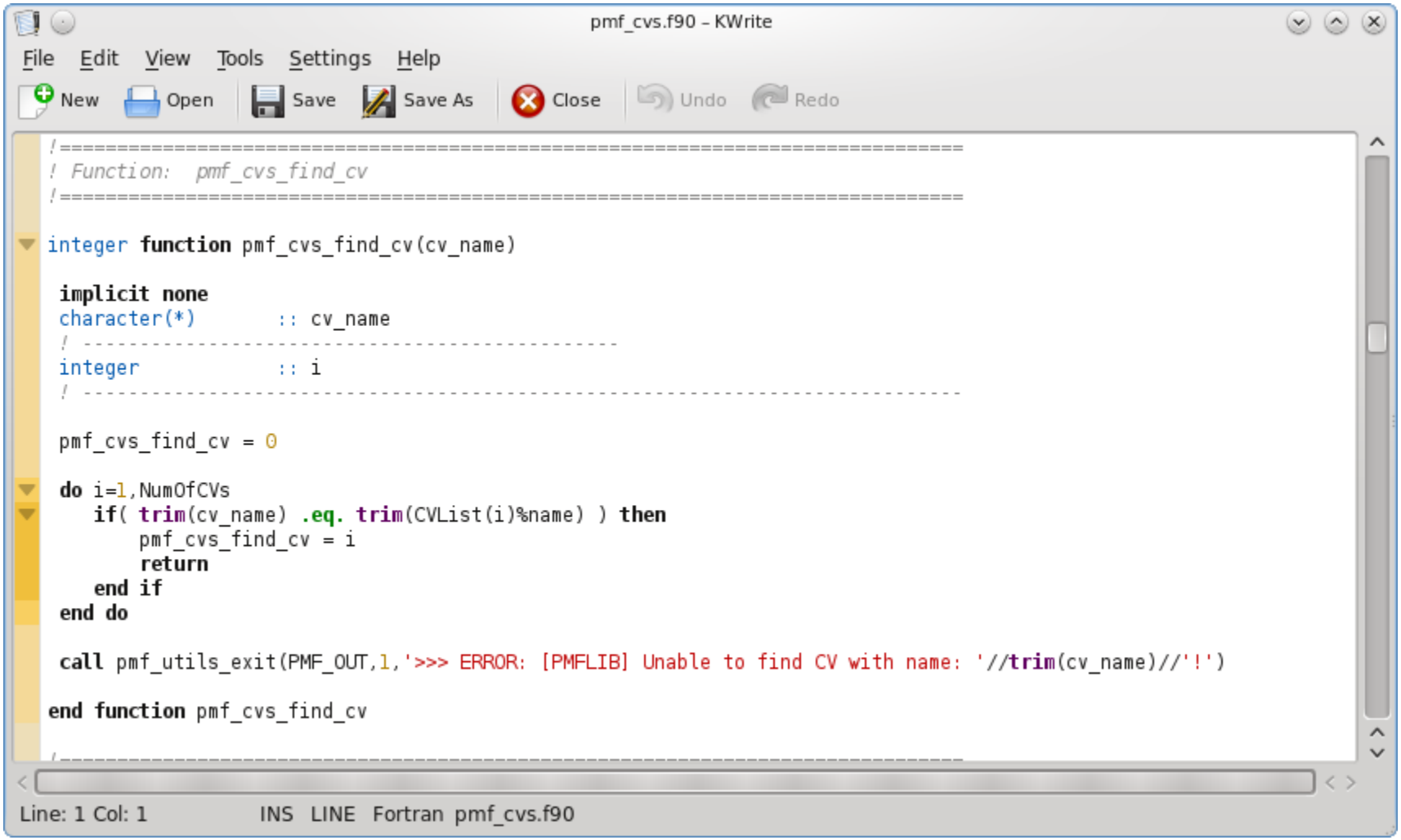


Přímočařejší ovládání – menu v dolní části napovídá možné akci. Pro volbu akce slouží kombinace nebo samostatná písmena

^písmeno – např. ^X je kombinace Ctrl + X

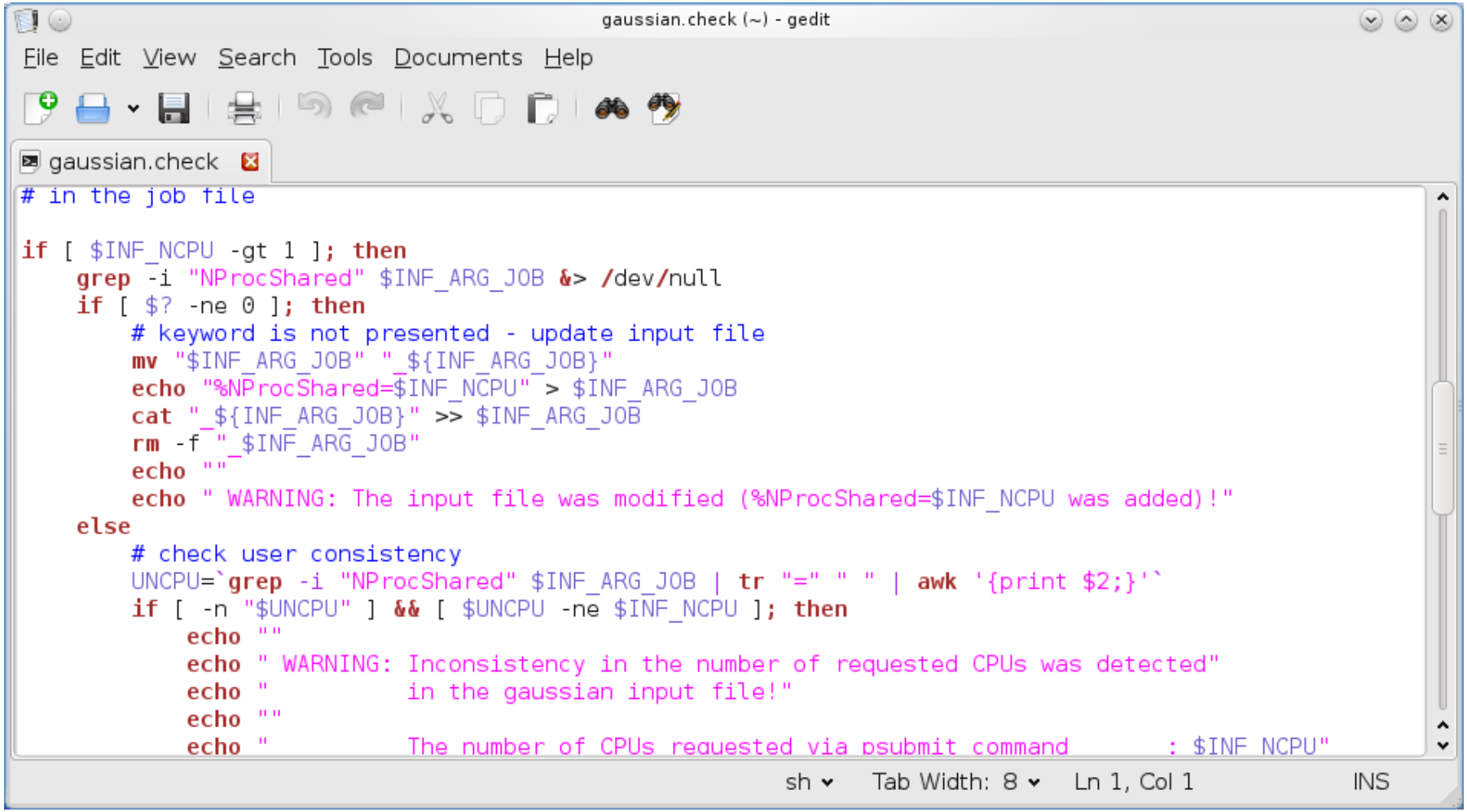
M-písmeno – např. M-M je kombinace Alt+M

kwrite



Rozšířená funkcionalita: **kate**

gedit


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

Cvičení

1. V editoru **vi** napište text, který bude obsahovat deset řádků. Na každém řádku budou dvě a více slov. Text uložte do souboru **mojedata.txt**
2. Příkazem **wc** ověřte, že soubor **mojedata.txt** má skutečně deset řádků.
3. Za použití rour(y) napište sekvenci příkazů, které na obrazovku vypíší pouze počet slov v souboru **mojedata.txt**
4. V grafickém textovém editoru (dle vašeho výběru) vytvořte soubor, který bude obsahovat **deset slov**, každé slovo na **novém** řádku. Text uložte do souboru **druha_data.txt**
5. Pomocí příkazu **paste** vytvořte soubor **vsechna_data.txt** , který bude obsahovat obsah souborů **mojedata.txt** a **druha_data.txt vedle sebe**.
6. Příkazem **wc** ověřte, že soubor **vsechna_data.txt** obsahuje právě deset řádků.
7. Soubor **vsechna_data.txt** otevřete v grafickém textovém editoru a jeho obsah **ověřte vizuálně**.
8. Otevřete soubor **vsechna_data.txt** v editoru **nano** a uložte pod jiným názvem ve formátu **mac**. Analyzujte oba soubory příkazem **wc** a vypište jejich obsah příkazem **cat**. Čím se liší vytvořený soubor od původního?

Procesy II

- příkazy
- spouštění úloh
- ukončovaní úloh

Příkazy

top	průběžně zobrazuje procesy setříděné podle zátěže procesoru (ukončení klávesou q)
ps	vypíše procesy běžící v daném terminálu nebo podle zadaných specifikací (`ps -u user_name`)
pstree	vypíše procesy (stromový výpis)
kill	zašle signál procesu, lze použít k ukončení problematických programů
nohup	spustí proces bez interakce s terminálem
sleep	spustí proces , který čeká po zadanou dobu
wait	čeká na dokončení procesů na pozadí
time	vypíše délku běhu procesu
ssh	spustí příkaz na vzdáleném počítači
jobs	vypíše procesy na pozadí
fg	převede proces do popředí
bg	převede proces do pozadí
disown	odpojí proces od terminálu

Spouštění příkazů a aplikací

Příkazy a systémové aplikace

```
$ ls -l
$ cp soubor.txt soubor1.txt
```

zadáváme pouze jméno příkazu nebo aplikace

příkaz

argumenty příkazu (mění chování příkazu nebo tvoří vstupní informace pro zpracování)

Uživatelské programy a skripty

```
$ ./muj_script
$ ~/bin/my_application
```

jméno programu nebo skriptu udáváme **včetně cesty** (absolutní nebo relativní)

Zrušení výpisu do terminálu

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

přesměrování výstupu uvádíme na konec příkazu (za argumenty)

Spouštění aplikací na pozadí

```
$ gimp &
```

na konec (za argumenty a přesměrování) příkazu uvedeme ampersand

Spouštění příkazů a aplikací, II

Terminál (užitečné klávesové zkratky):

Ctrl+C běžícímu procesu zašle signál SIGINT (Interrupt), proces je ve většině případů násilně ukončen

Ctrl+D zavře vstupní proud spuštěného procesu

Ctrl+Z pozastaví běh procesu, další osud procesu lze kontrolovat pomoci příkazů **bg**, **fg**, **disown**

Kde se nachází systémový příkaz:

type vypíše cestu k systémovému příkazu nebo programu

Příklady:

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

$ type pwd
pwd is a shell builtin
```

příkaz pwd je implementován jako vnitřní příkaz shellu

Příklady

```
$ 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   # ukončí aplikaci kwrite

$ kwrite      # spustí aplikaci kwrite na popředí
^Z            # pozastaví aplikaci
[1]+  Stopped                 kwrite
$ jobs        # vypíše aplikace na pozadí nebo
              # pozastavené aplikace
[1]+  Stopped                 kwrite
$ bg 1        # pozastavenou aplikaci 1 spustí na pozadí
[1]+ kwrite &
$ jobs
[1]+  Running                 kwrite &
```

Cvičení

1. Prakticky si vyzkoušejte příklady uvedené na předchozí stránce.
2. Změřte délku běhu procesu **sleep 0.003**, jak dlouho program běží? Proveďte celou řadu experimentů s různou hodnotou času v rozmezí od 0.0001 do 10 sekund. Zdůvodněte pozorované rozpory pro malé hodnoty času.
3. Zjistěte jméno procesu s číslem **1**, jaký uživatel je jeho vlastníkem?
4. Zkuste tento proces ukončit. Z jakého důvodu není operace povolena?
5. Spusťte program **VMD**. Jaký má jeho proces PID? Program násilně ukončete příkazem **kill**.