C2110

Operační systém UNIX a základy programování

3. lekce

Petr Kulhánek

kulhanek@chemi.muni.cz

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

Obsah

- **Souborový systém II**

  vytváření adresářů, kopírování souborů a adresářů, přesouvání souborů a adresářů, mazaní souborů a adresářů, speciální znaky, kvóty

- **Procesy**

  procesy, standardní vstup a výstup, přesměrování, roury, úvod do proměnných, manipulace s procesy

Souborový systém II

- Vytvoření adresářů
- Kopírovaní souborů a adresářu
- Přesouvání souborů a adresářů
- Mazání souborů a adresářů
- Speciální znaky
- Kvóty

Vytvoření adresářů

- **Vytvoření adresáře**

  $ mkdir jmeno_adresare

- **Vytvoření vnořených adresářů**

  $ mkdir **-p** jmeno_adresare1/jmeno_adresare2/jmeno_adresare3

Kopírování

Ke kopírování slouží příkaz “cp”

$ cp soubor1 soubor2

vytvoří kopii souboru “soubor1” s názvem “soubor2”

$ cp soubor1 soubor2 soubor3 adresar1/

kopíruje soubory “soubor1”, “soubor2”, “soubor3” do adresáře “adresar1”

$ cp **-r** adresar1 adresar2

vytvoří kopii adresáře “adresar1” s názvem “adresar2”; pokud adresář “adresar2” již existuje, výtvoří kopii adresáře “adresar1” jako podadresář adresáře “adresar2”

$ cp **-r** soubor1 adresar2 soubor3 adresar1/

kopíruje soubory “soubor1”, “soubor3” a adresář “adresar2” do adresáře “adresar1”

Přesouvání

K přesouvání nebo přejmenovávání slouží příkaz “mv”

$ mv soubor1 soubor2

přejmenuje soubor “soubor1” na “soubor2”

$ mv soubor1 soubor2 soubor3 adresar1/

přesune soubory “soubor1”, “soubor2”, “soubor3” do adresáře “adresar1”

$ mv **-r** adresar1 adresar2

přejmenuje adresář “adresar1” na “adresar2”; pokud adresář “adresar2” již existuje, přesune adresář “adresar1” do adresáře “adresar2”

$ mv **-r** soubor1 adresar2 soubor3 adresar1/

přesune soubory “soubor1”, “soubor3” a adresář “adresar2” do adresáře “adresar1”

Speciální znaky

Speciální znaky v názvech souborů:

- ***** - cokoliv v názvu souboru (bez skrytých souborů)
- **?** - jeden znak v názvu souboru
- **[]** - rozsah (jeden znak) v názvu souboru, př. [ajk], [a,j,k], [a-j]

Expanzi specálních znaků provádí shell ještě **před spuštěním** samotného příkazu. Expanzi lze zabránit uvedením jména v uvozovkách nebo použitím zpětného lomítka před speciálním znakem.

Příklady:

$ rm *

smaže všechny soubory v aktuálním adresáři (kromě adresářů)

$ mv A? Tmp/

přesune soubory s názvem začínajícím písmenem “A” a obsahujícím dva znaky do adresáře “Tmp”

Speciální znaky

K expanzi speiálních znaků dojde pouze tehdy, pokud exisuje alespoň jeden soubor vyhovující danému předpisu:

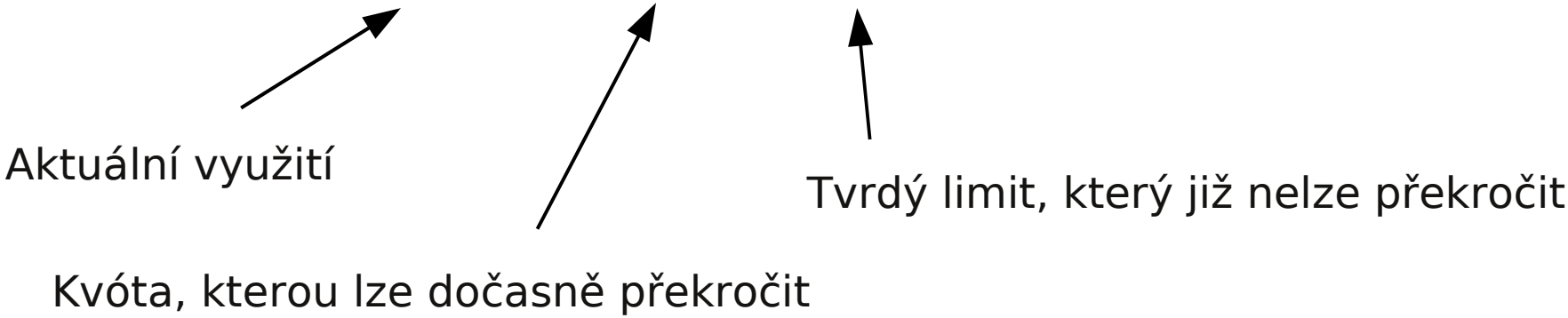
Příklady:

$ cd
$ echo D*
Desktop Documents Downloads
$ echo A*
A*
$ echo “D*”
D*
$ echo D*
D*

Kvóty

Pro vaše domovské adresáře jsou nastaveny kvóty na využití diskového prostoru na diskovém oddílu `/dev/mapper/server1-home`. Aktuální stav zaplnění a nastavení kvót lze zjistit příkazem **quota**:

```
[kulhanek@wolfn ~]$ quota -vs
Disk quotas for user kulhanek (uid 18773):
     Filesystem  blocks   quota   limit   grace   files   quota   limit
/dev/mapper/server1-home
                  1550M   1954M   2051M           20453       0       0
```

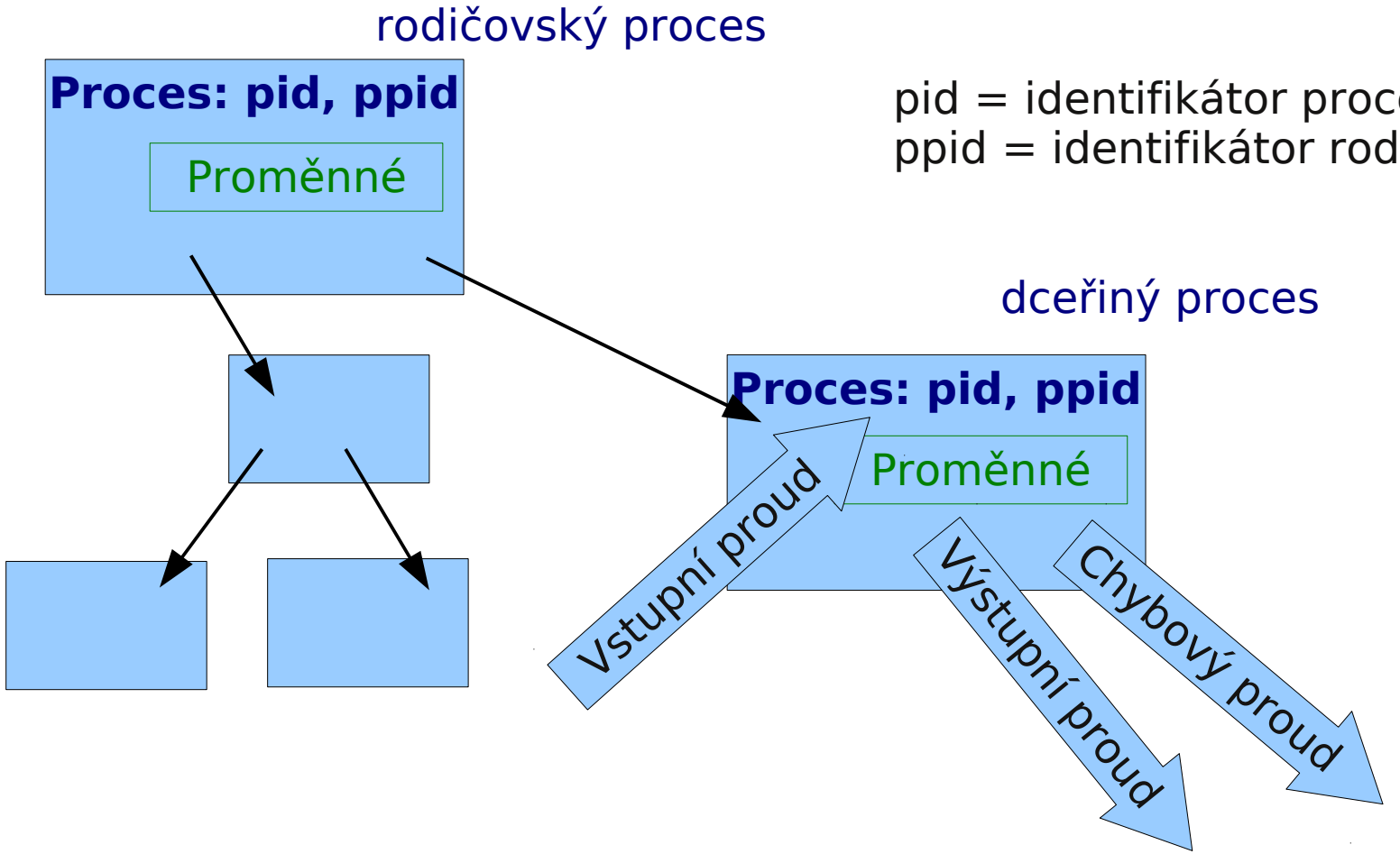


Překročení kvóty může vést k **nefunkčnímu přihlášení** pomocí grafického rozhraní. V tomto případě se přihlašte v textovém terminálu (např: Ctrl+Alt+F1) a přesuňte soubory na jiný diskový oddíl (např. dočasně do adresáře /scratch/vas_login nebo smažte nepotřebné soubory.

Procesy

- Procesy
- Standardní proudy
- Přesměrování standardních proudů
- Roury
- Proměnné
- Manipulace s procesy

Procesy

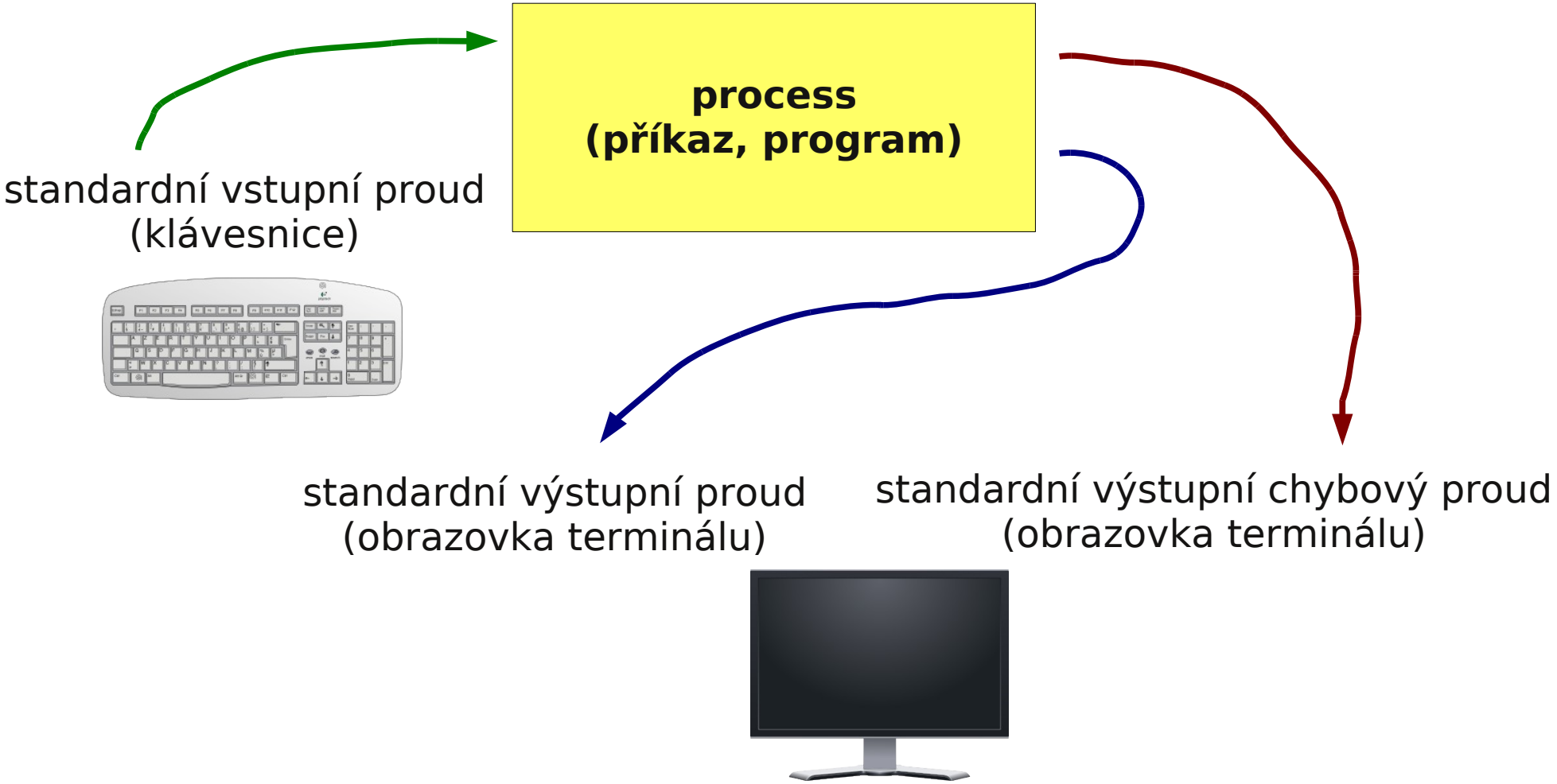


- Prvním spuštěným procesem po spuštění systému je proces “**init**”
- Každý příkaz spuštený v shellu (příkazové řádce) je procesem

Standardní proudy

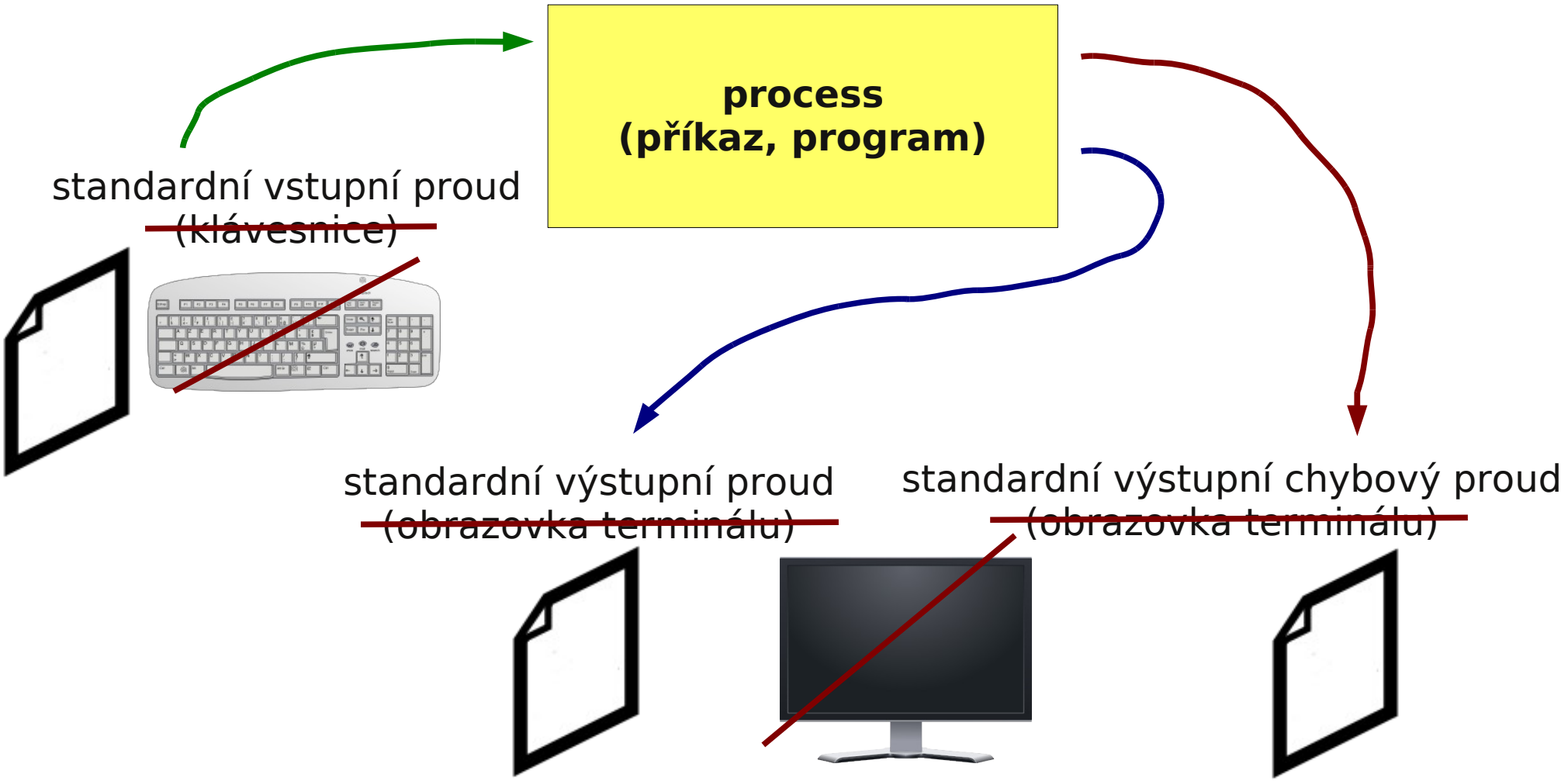
Vstupně-výstupní proudy slouží procesu ke **komunikaci** se svým okolím.

Každý proces otevírá **tři standardní proudy**:



Přesměrování

Vstupně-výstupní proudy lze přesměrovat tak, aby používaly soubory místo klávesnice či obrazovky.



Přesměrování vstupu

Přesměrování standardního vstupu programu my_command ze souboru input.txt.

```
$ my_command < input.txt
```

Přesměrování standardního vstupu programu my_command ze souboru skriptu.

```
.......
./my_command << EOF
prvni radka textu
druha radka textu
treti radka textu
EOF
......
```

značka určující konec vstupu (volí uživatel)

text, který tvorí načítaný vstup

konec vstupu, *značku* ***nesmí*** *obklopovat mezery*

Tento způsob přesměrování je obzvláště výhodné používat ve skriptech, nicméně funguje i v příkazové řádce. Výhodou je expanze proměnných v načítaném textu.

Přesměrování výstupu

Přesměrování standardního výstupu programu my_command do souboru output.txt. (Soubor output.txt je vytvořen. Pokud již existuje, je jeho původní obsah **smazán**.)

```
$ my_command > output.txt
```

Přesměrování standardního výstupu programu my_command do souboru output.txt. (Soubor output.txt je vytvořen. Pokud již existuje, je výstup programu my_command **připojen** na jeho konec.)

```
$ my_command >> output.txt
```

Podobná pravidla platí pro standardní chybový výstup, v tomto případě se používají nasledující operátory:

```
$ my_command 2> errors.txt
$ my_command 2>> errors.txt
```

Spojování výstupních proudů

Standardní výstup **a** standardní chybový výstup programu my_command lze současně přesměrovat do souboru output.txt.

```
$ my_command &> output.txt
```

Výše uvedený postup nelze použít pro operátor >>.

~~$ my_command &>> output.txt~~ nefunguje

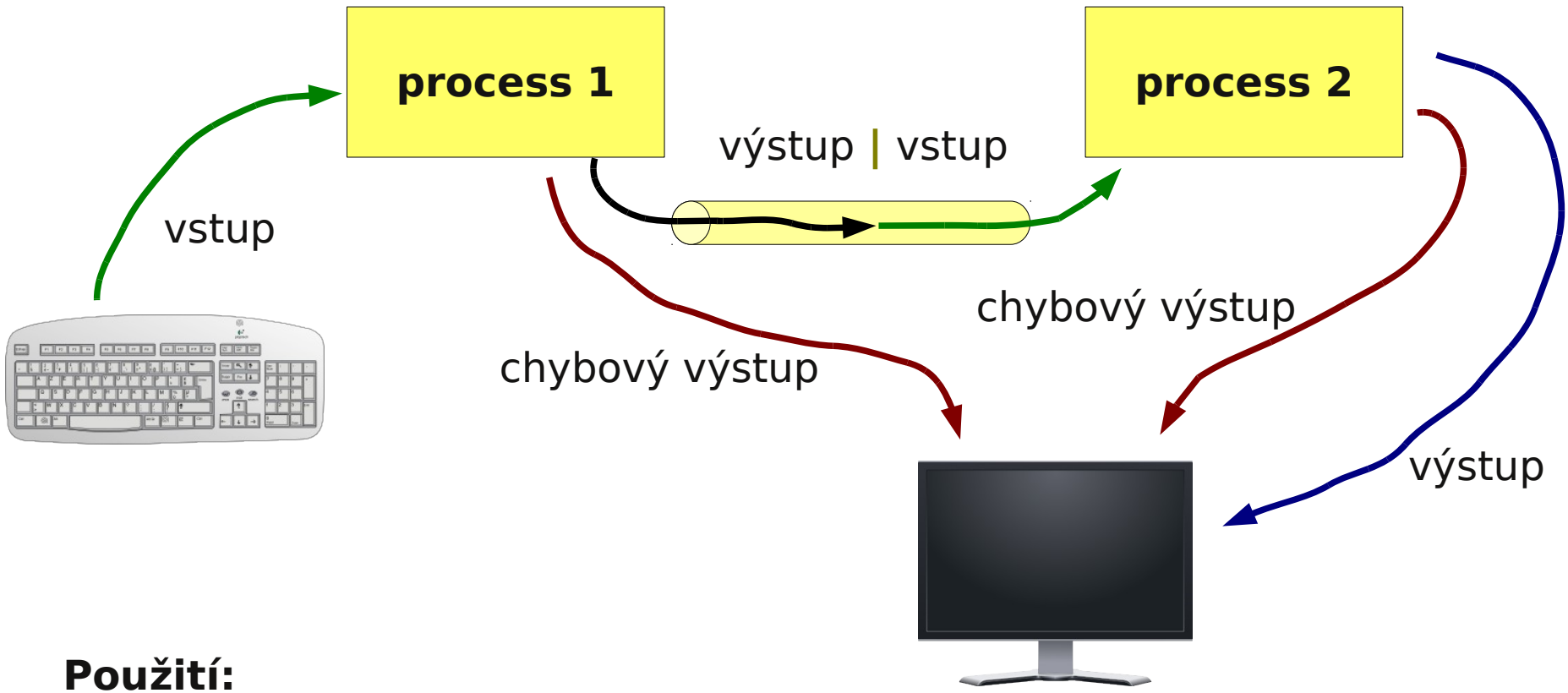
Řešení: Nejdříve je nutné **přesměrovat** standardní výstup a poté **spojit** standardní chybový výstup s výstupem standardním.

```
$ my_command >> output.txt 2>&1
```
pořadí je důležité!

~~$ my_command 2>&1 >> output.txt~~ nefunguje

Roury (pípy)

Roury slouží ke spojování standardního výstupu jednoho procesu se standardním vstupem jiného procesu.

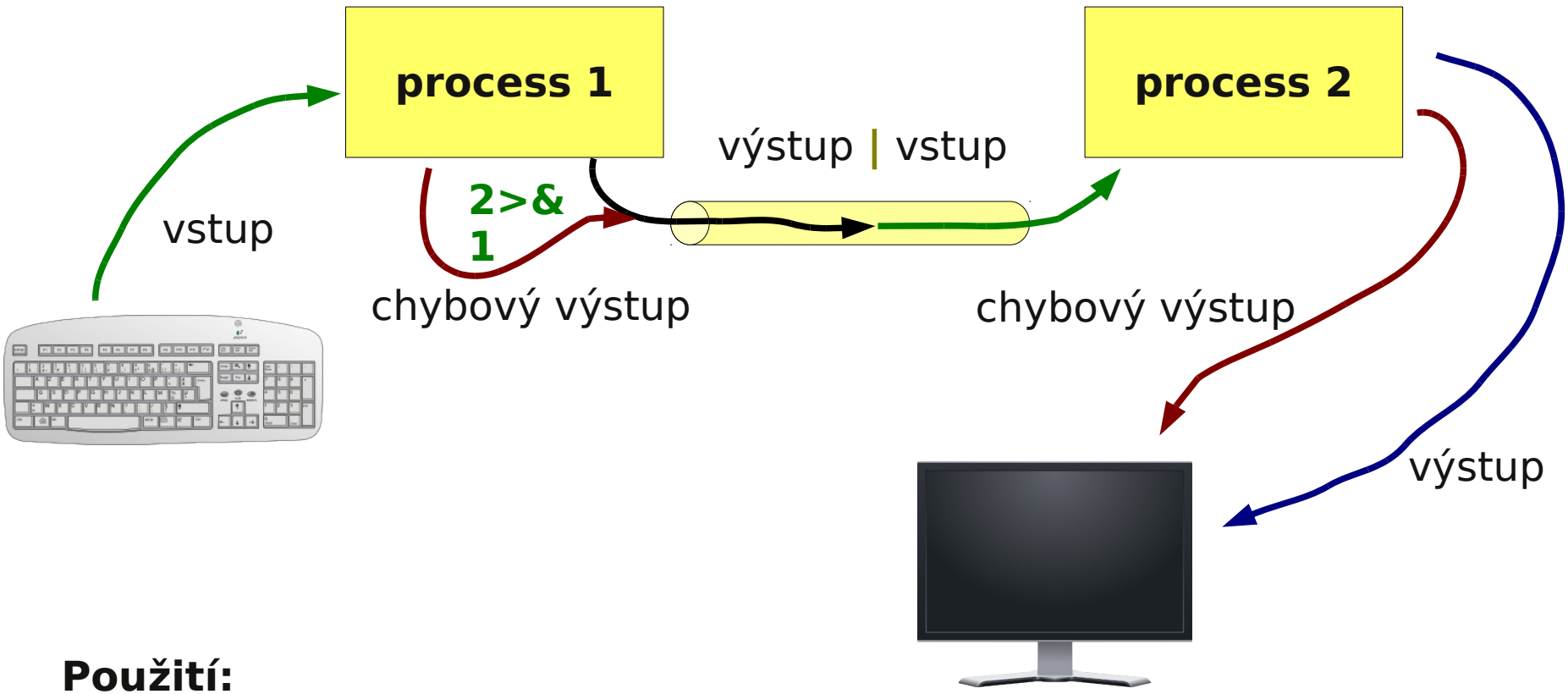


Použití:

$ command_1 | command_2

Roury a chybový proud

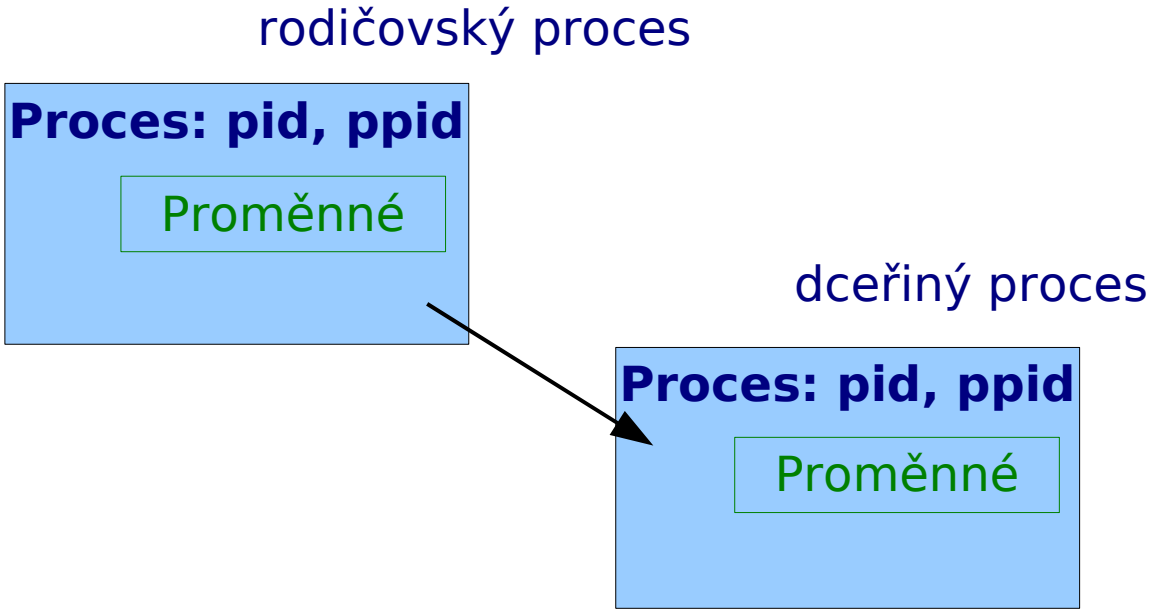
Přenos standardního chybového výstupu přes rouru je možné provést po jeho spojení se standardním výstupem.



Použití:

$ command_1 **2>&1** | command_2

Proměnné



Dceřiný proces dědí proměnné rodičovského procesu, které jsou exportované.

nesmí být mezera mezi jménem a =

Nastavení proměnné:

```
$ JMENO_PROMENNE=hodnota
$ JMENO_PROMENNE=”hodnota s mezerami”
```

Export proměnné:

```
$ export JMENO_PROMENNE
```

Přístu k proměnné:

```
$ echo $JMENO_PROMENNE
```

Proměnné

Přehled nastavených proměnných:

$ set

Odstranění proměnné:

$ unset JMENO_PROMENNE

Expanze proměnných se provádí před spuštěním příkazu. Expanzi lze zabránit pomocí jednoduchých uvozovek nebo zpětného znaku před znakem $.

Příklady:

```
$ A=5
$ echo $A
5
$ echo “$A”
5
$ echo \$A
$A
$ echo '$A'
$A
```

Manipulace s procesy

Procesy:

- top — informace o běžících procesech v systému
- ps — vypíše informace o běžících procesech v systému
- kill — zašle signál procesu
- nohup — spustí process bez interakce s terminálem
- time — vypíše délku běhu procesu
- wait — čeká na dokončení procesů na pozadí
- ssh — spustí příkaz na vzdáleném počítači

Spouštění příkazů

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

Spouštění na pozadí (znak & na konci příkazu):

```
$ kwrite &
```

Kde se nachází spuštěný příkaz:

Proměnná PATH

Příkaz type

Spouštění příkazů ...

$ command

1. Nejdříve se příkaz hledá v tabulce s již použitými příkazy

```
$ hash
hits    command
1    /bin/rm
3    /bin/ls
```

Tabulku lze smazat příkazem:
$ hash -r

2. Pokud není příkaz nalezen, hledá se v adresářích uvedených v systémové proměnné PATH

```
$ echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:
/sbin:/bin:/usr/games
```

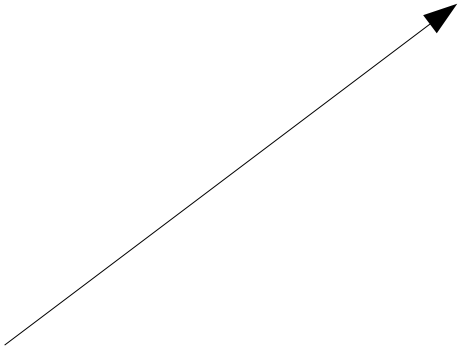
pořadí prohledávání

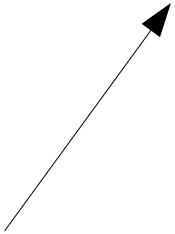
Adresáře se oddělují znakem : (dvojtečka)

Spouštění příkazů ...

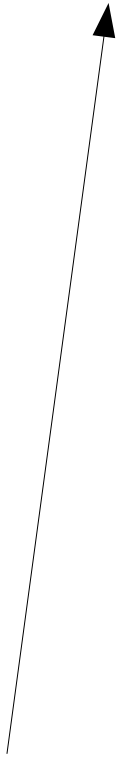
Změna proměnné PATH

```
$ export PATH=/moje/cesta/k/mym/prikazum:$PATH
```





oddělující znak



Cesta k adresáři obsahující příkazy, u kterých chci, aby byly přístupné bez uvádění cesty.

Cesta se vždy uvádí absolutně! (uvádění relativních cest je bezpečnostním rizikem)

Původní hodnota proměnné **PATH**
(nutné pro nalezení sytémových příkazů)