evropský sociální fond v ČR
EVROPSKÁ UNIE
MINISTERSTVO ŠKOLSTVÍ, MLÁDEŽE A TĚLOVÝCHOVY
OP Vzdělávání pro konkurenceschopnost
INVESTICE DO ROZVOJE VZDĚLÁVÁNÍ

PSK3-5

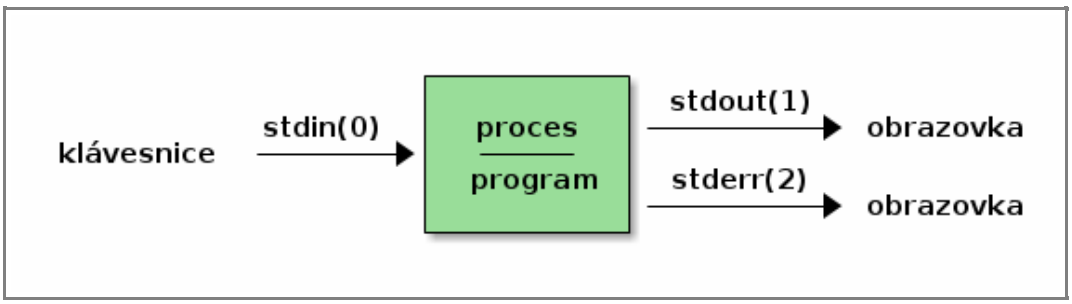
Název školy:	Vyšší odborná škola a Střední průmyslová škola, Božetěchova 3
Autor:	Ing. Marek Nožka
Anotace:	Standardní vstup, výstup a jejich přesměrování
Vzdělávací oblast:	Informační a komunikační technologie
Předmět:	Počítačové sítě a komunikační technika (PSK)
Tematická oblast:	Operační systém Linux/Unix
Výsledky vzdělávání:	Žák přesměrovává vstupy/výstupy z/do souboru, používá roury
Klíčová slova:	Linux, Unix, shell, roura, pipe, stdin, stdout, stderr
Druh učebního materiálu:	Online vzdělávací materiál
Typ vzdělávání:	Střední vzdělávání, 4. ročník, technické lyceum
Ověřeno:	VOŠ a SPŠE Olomouc; Třída: 4L
Zdroj:	Vlastní poznámky, Vilém Vychodil: Linux Příručka českého uživatele

Přesměrování vstupu a výstupu

Vstup a výstup

Jedním ze základních stavebních kamenů operačního sytému Unix je následující přístup:

- Každý program se chová jako filter.
- Vždy se předpokládá, že výstup jednoho programu se může stát vstupem druhého programu.
- Univerzálním komunikačním rozhraním je text.

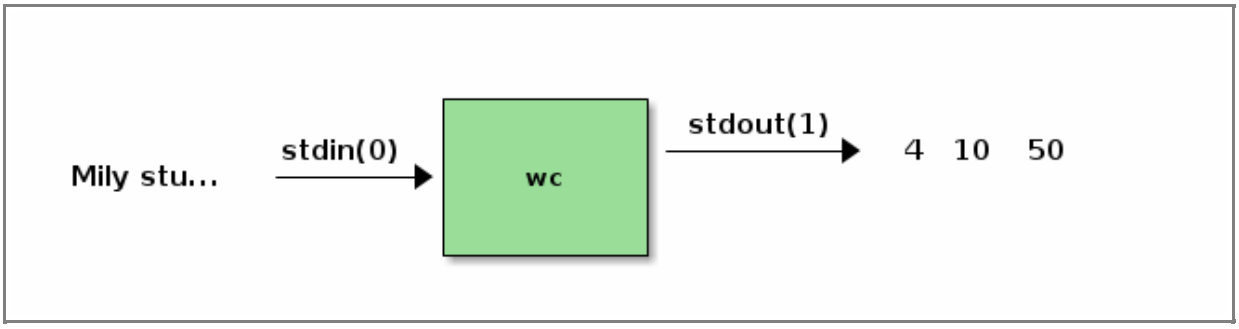


V Unixu má každý proces (program) standardní vstup **stdin**, standardní výstup **stdout** a standardní chybový výstup **stderr**. Vstup standardně čte z klávesnice vždy, když uživatel stiskne klávesu `Enter`. Výstup a chybový výstup se během běhu programu vypisuje na terminál, tedy na obrazovku.

```
$ wc
Mily studente,

dnes se ucime o OS Unix
Hezky den
^D
>>>>   4      10      50
```

Například program `wc` čte text zadaný z klávesnice a po jeho ukončení (*klávesová zkratka Ctrl+D vkládá znak konce souboru*) vypíše počet znaků, slov a řádků.



Všechny programy ale nutně nemusí *stdin* používat. Například program `ls`, který vypisuje obsah adresáře žádný standardní vstup nečte.

Přesměrování výstupu do souboru

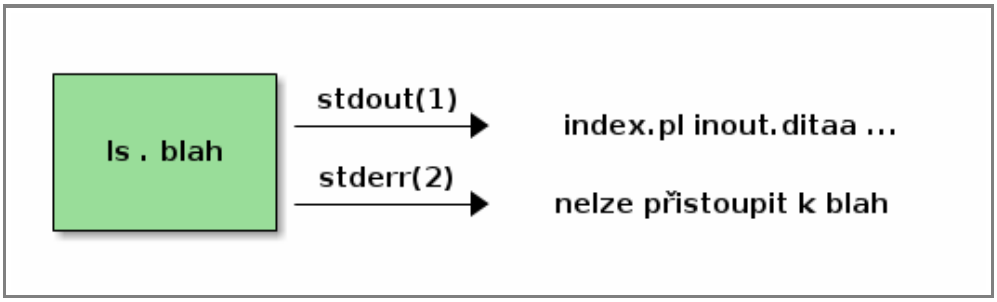
Výstup každého programu je možné přesměrovat do souboru pomocí metaznaku `>`:

`>SOUBOR` standardní výstup je přesměrován do souboru
`>>SOUBOR` standardní výstup je připojen na konec souboru
`2>SOUBOR` standardní chybový výstup je přesměrován do souboru
`2>>SOUBOR` standardní chybový výstup je připojen na konec souboru
`&>SOUBOR` oba standardní výstupy jsou přesměrovány do souboru

Například:

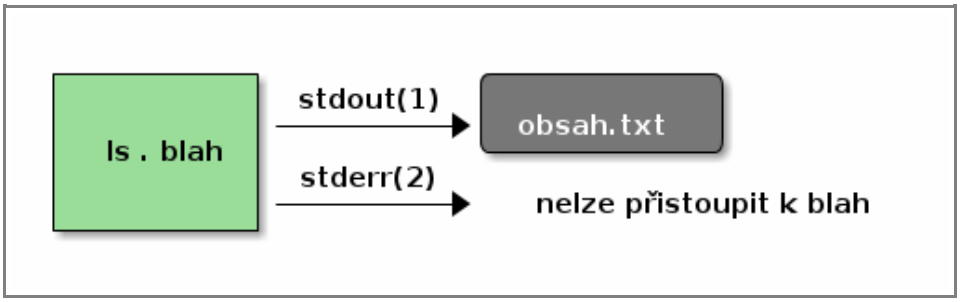
```
$ ls . blah
>>>>ls: nelze přistoupit k blah: Adresář nebo soubor neexistuje
>>>>.:
>>>>index.pl  inout.ditaa  inout.png
```

Program vypíše obsah aktuálního adresáře a adresáře `blah`. Protože ale adresář `blah` neexistuje program vypíše na chybový výstup chybové hlášení.



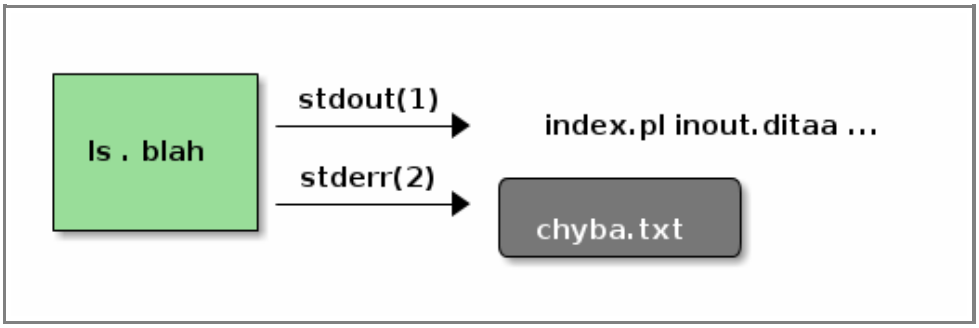
Nyní výstup přesměrujeme do souboru `obsah.txt`. Chybové hlášení je stále vidět na terminálu (obrazovce), ale výpis adresáře se provedl do souboru:

```
$ ls . blah >obsah.txt
>>>>ls: nelze přistoupit k blah: Adresář nebo soubor neexistuje
```



Jestliže přesměrujeme do souboru pouze chybový výstup, situace se otočí:

```
$ ls . blah 2>chyba.txt
>>>>.:
>>>>index.pl  inout.ditaa  inout.png
```



Obsah souboru můžeme vypsat pomocí příkazu `cat`:

```
$ cat obsah.txt chyba.txt
>>>>.:
>>>>index.pl
>>>>inout.ditaa
>>>>inout.png
>>>>obsah.txt
>>>>/bin/ls: nelze přistoupit k blah: Adresář nebo soubor neexistuje
```

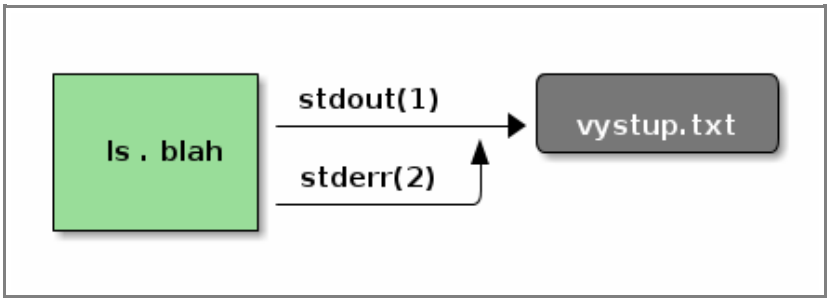
Při přesměrování lze místo jména SOUBORu použít souborový deskriptor. Ten se zadává ve formátu `&CISLO`, kde CISLO označuje deskriptor.

deskriptor	význam
0	standardní vstup
1	standardní výstup
2	standardní chybový výstup

Příkaz:

```
$ ls . blah >>vystup.txt 2>&1
```

Přesměruje standardní výstup na konec souboru a zároveň přesměruje chybový výstup na standardní výstup, takže se v souboru `vystup.txt` objeví *stdout(1)* i *stderr(2)*.



Přesměrování vstupu ze souboru

Vstup programu lze přesměrovat pomocí metaznaku <. Proces (program) potom nebude číst vstupní data z klávesnice ale ze souboru.

Jako příklad si uvedeme krátký program v jazyce Python, který provede součet všech čísel zadaných na vstup.

```
 1 #!/usr/bin/python
 2 # -*- coding: utf8 -*-
 3 # Soubor:   soucet.py
 4 # Licence:  GNU/GPL
 5 # Popis:    Program provede součet všech čísel, které přečte z stdin
 6 ################################################################################
 7 import sys
 8 from sys import stdin, stdout, stderr
 9 ################################################################################
10 soucet=0
11 while True:
12     try:
13         line = raw_input()
14         cisla = line.split()
15         for c in cisla:
16             c=float(c)
17             soucet+=c
18     except EOFError:
19         print soucet
20         exit(0)
21     except KeyboardInterrupt:
22         stderr.write("Program prerusen uzivatelem\n")
23         exit(1)
24     except:
25         stderr.write("ERROR: '{}' neni cislo\n".format(c))
26         exit(2)
```

`--> stáhnout

```
./soucet.py
1 2 3
4 5
>>>>15.0
```

Čísla, která se mají sečíst můžeme zapsat do souboru `secti to. txt`:

```
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21
22 23 24 25 26 27 28 29 30 31 32 33 34 35 36
37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56
57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72
73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89
90 91 92 93 94 95 96 97 98 99 100
```

`--> stáhnout

Potom stačí jednoduše zavolat:

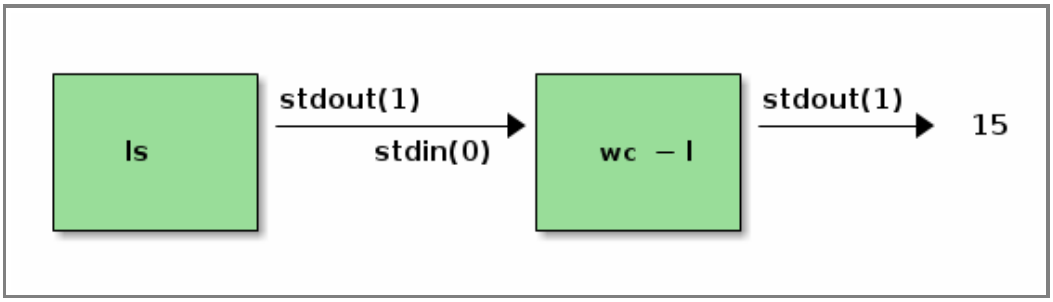
```
./soucet.py <secti to. txt
>>>>5050.0
```

Roury

Často je velice užitečné spojit dva programy a výstup jednoho přesměrovat na vstup druhého. Slouží k tomu mechanizmus, který se označuje jako roura (pipe). Pro zřetězení několika programů pomocí roury se používá metaznak `|` .

```
ls *.txt | wc -l
>>>> 15
```

Výstup programu `ls` je přesměrován na vstup programu `wc`, který počítá řádky. Výsledkem je tedy počet souborů s příponou `. txt`.



Takto lze spojovat do kolony i více programů. Například:

```
ls -l *.png | cut -d ' ' -f 5,10 | sort -n
>>>>3674 roura.png
>>>>3842 wc.png
>>>>5091 oba.png
>>>>5123 blah.png
>>>>5909 obsah.png
>>>>5947 chyba.png
>>>>6417 inout.png
```

Program `ls` podrobně vypíše soubory s příponou `.png`, program `cut` vybere z výpisu jen dva sloupce a program `sort` je seřadí podle velikosti.

Rouru není možné použít vždy. Je to například pokud spolu chtějí komunikovat dva procesy, které spouští vždy jiný uživatel. V těchto případech je možné použít pojmenovanou rouru. Tu je třeba nejprve vytvořit a poté je možné s ní pracovat jako by to byl soubor.

```
$ mkfifo /tmp/mojetrubka
$ chmod a+r /tmp/mojetrubka
$ ls -l *.png | cut -d ' ' -f 5,10  >/tmp/mojetrubka
```

Jiný uživatel potom může z roury číst

```
$ sort -n </tmp/mojetrubka
```