Pokročilá práce s terminálem

15.02.2025

Operační systémy
IV. ročník

Bc. Lukáš Pospíšil

MěSOŠ
Klobouky u Brna

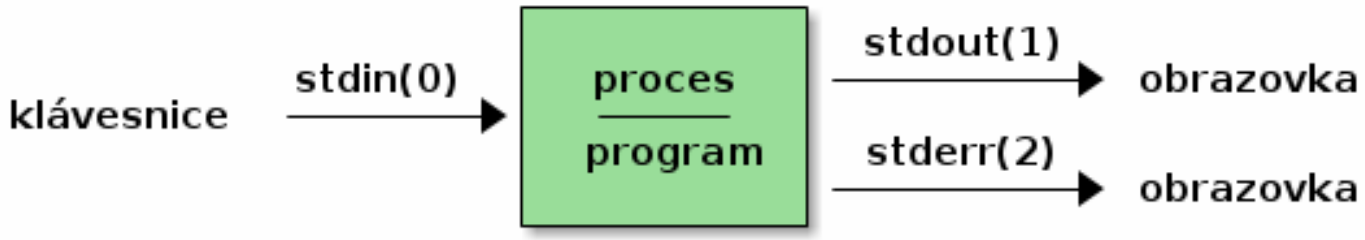
Cíl tématu

- **Znám** pojmy:
  - datový proud, stdin, stdout, stderr, pipe, filtr, skript, cron, at
- **Popíšu** a **vyjmenuju:**
  - operátory pro přesměrování datových proudů
  - operátory při práci s shellovými skripty
- **Umím:**
  - využívat roury pro propojení výstupů a vstupů příkazů
  - používat filtry pro manipulaci s textovými daty
  - vytvářet jednoduché shellové skripty
  - Využívat plánování úloh pomocí cron a at

PŘESMĚROVÁNÍ

Přesměrování

- Pro **manipulaci se standardními vstupy a výstupy**
- Využíváme shell
- Základní datové proudy
  - **stdin**
  - **stdout**
  - **stderr**

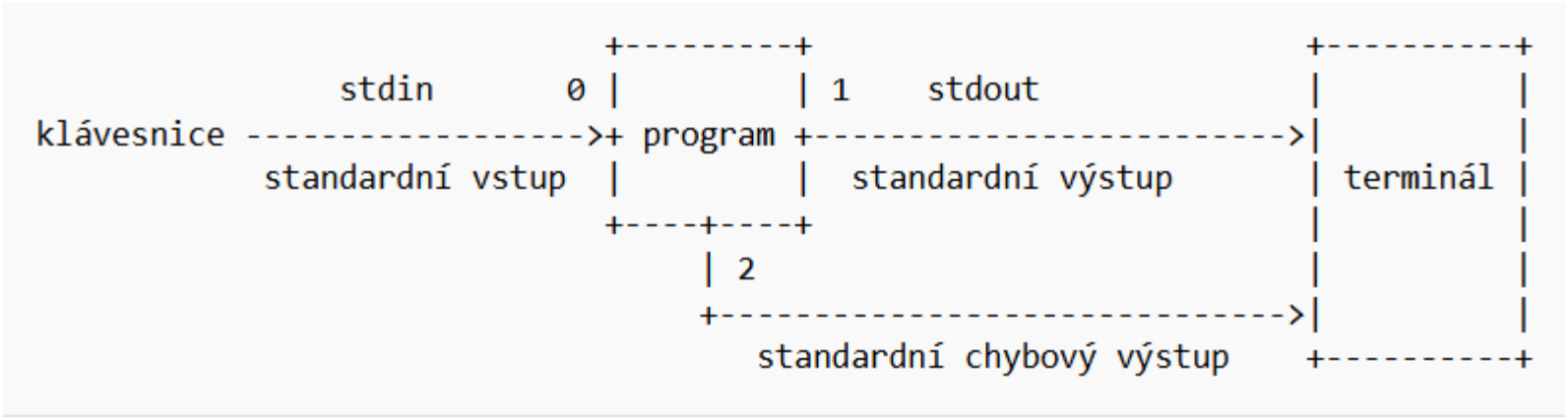


Základní datové proudy

- **Standardní vstup (číslo 0) – stdin**
  - vstup od uživatele nebo z jiného zdroje
- **Standardní výstup (číslo 1) – stdout**
  - výstup úspěšně provedených příkazů
- **Standardní chybový výstup (číslo 2) – sderr**
  - výstup chybových hlášení

Základní datové proudy – jak fungují

- Program potřebuje **získat data** – čte z proudu **0**
  - potřebuje **vypsat data** – použije proud **1**
  - potřebuje **vypsat chybovou hlášku** – použije proud **2**
- Pomocí oddělení **odlišujeme výstup** výsledku programu a chybových hlášení



Symboly pro přesměrování

Konstrukce	Funkce
`>`	Přesměrování standardního výstupu (vytvoří nebo přemaže existující soubor)
`>>`	Přesměrování standardního výstupu (vytvoří nebo přidá za konec existujícího souboru)
`<`	Přesměrování standardního vstupu
`2>`	Přesměrování do standardního chybového výstupu
`2>&1`	Spojení standardního chybového výstupu se standardním výstupem
`1>&2`	Spojení standardního výstupu se standardním chybovým výstupem (používá se při programování skriptů)
`<<MARK`	Here document (používá se při programování skriptů)

Datové proudy

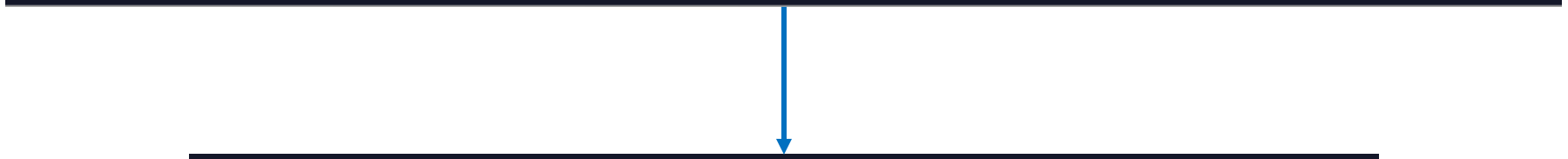
- Program potřebuje **získat data** – čte z proudu **0**
  - potřebuje **vypsat data** – použije proud **1**
  - potřebuje **vypsat chybovou hlášku** – použije proud **2**
- Pomocí oddělení **odlišujeme výstup** výsledku programu a chybových hlášení

Přesměrování výstupu

- Výstup pro daný program do souboru
- Pomocí operátorů
  - **>** - **přepíše** obsah souboru, existuje-li
  - **>>** - **přilepí** obsah soubor nakonec
- Můžeme přesměrovat i soubor do zařízení
  - cat hudba.mp3 > dev/audio

Přesměrování výstupu

```
root@mesosdev:/etc/apt# ls
apt.conf.d   keyrings       preferences.d.save  sources.list.curtin.old  trusted.gpg
auth.conf.d  preferences.d  sources.list        sources.list.d           trusted.gpg.d
root@mesosdev:/etc/apt# ls > vypis.txt
root@mesosdev:/etc/apt#
```



```
 GNU nano 6.2                                    vypis.txt
apt.conf.d
auth.conf.d
keyrings
preferences.d
preferences.d.save
sources.list
sources.list.curtin.old
sources.list.d
trusted.gpg
trusted.gpg.d
vypis.txt
```

Přesměrování vstupu

- Pomocí operátoru **<**
- Vstup může být např. txt soubor
- **Můžeme i přesměrovat výstup jiného programu**
  - tzn. výstup jednoho programu může být vstupem druhé programu

```
root@mesosdev:/etc/apt# head < vypis.txt
apt.conf.d
auth.conf.d
keyrings
preferences.d
preferences.d.save
sources.list
sources.list.curtin.old
sources.list.d
trusted.gpg
trusted.gpg.d
root@mesosdev:/etc/apt#
```

```
root@mesosdev:/etc/apt# head < vypis.txt > druhy_vypis.txt
root@mesosdev:/etc/apt# cat druhy_vypis.txt
apt.conf.d
auth.conf.d
keyrings
preferences.d
preferences.d.save
sources.list
sources.list.curtin.old
sources.list.d
trusted.gpg
trusted.gpg.d
```

Přesměrování chybového výstupu

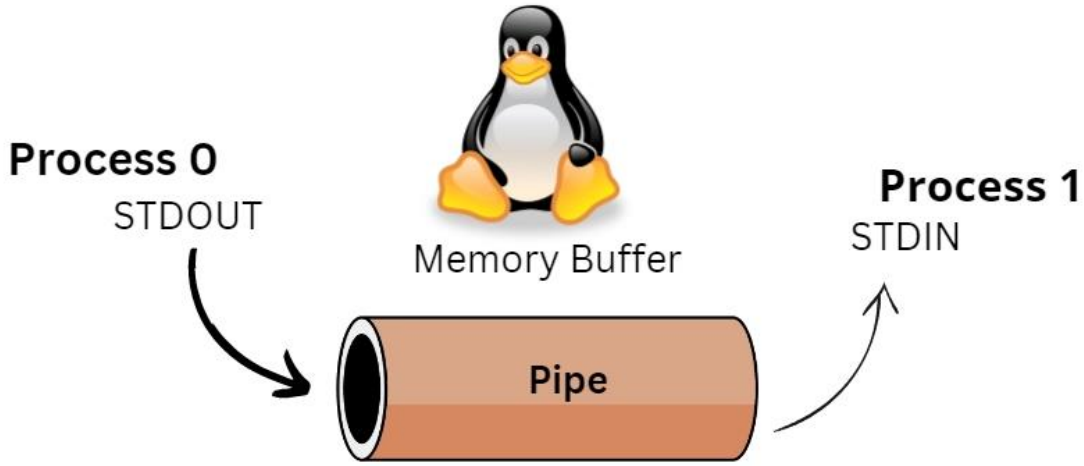
- Přesměrování **chyb** z obrazovky do jiného souboru
  - nějaký log soubor
  - využívá se při provádění skriptů
- Pomocí operátoru **2>**

```
root@mesosdev:/etc/apt# ls neexistujici_adresar 2> chyba.log
root@mesosdev:/etc/apt# cat chyba.log
ls: cannot access 'neexistujici_adresar': No such file or directory
root@mesosdev:/etc/apt#
```

ROURY

Roury

- Anglicky **pipes**
- **Propojení výstupu jednoho programu se vstupem druhého příkazu**
- Využití při složitějších operacích
  - neukládáme mezivýsledky do souboru



MěSOŠ Klobouky u Brna

Roury - syntaxe

- Používáme symbol **|**
- Syntaxe:
  - příkaz1 | příkaz2
    - výstup prvního příkazu je okamžitě vstup druhého příkazu
- Například: **ls | grep ".txt" | wc –l**
  - pro zjištění textových souborů v adresáři
  - nejdříve zobrazíme obsah adresáře pomocí ls
  - pomocí grep vybereme řádky s příponou txt
  - a s wc-l spočítáme počet řádků

Roury – příklady použití na školním serveru na API

- Počet procesů související s Pythonem
  - ps aux zobrazí všechny procesy běžící v systému
  - grep „python“ vyhledá řádky obsahující python
  - wc –l spočítá počet řádků

```
root@mesosdev:/home/mesosdev-api/htdocs/api.mesosdev.cz/mesosAPI# ps aux | grep
"python" | wc -l
9
```

Roury – příklady použití na školním serveru na API

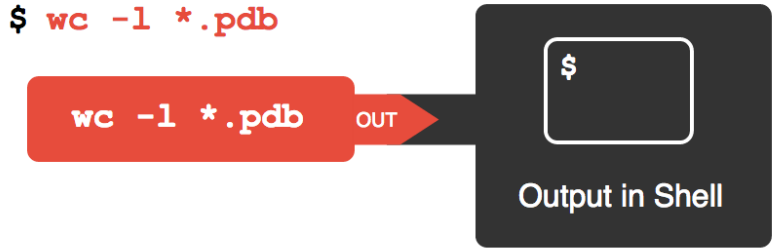
- Deset nejvyskytovanějších slov v souboru main.py
  - cat přečte obsah souboru
  - tr nahradí mezery novými řádky (takže co slovo, to řádek)
  - sort seřadí řádky abecedně
  - uniq –c odstraní duplicitní řádky
    - s přepínačem –c spočítá, kolikrát se řádek opakuje
  - sort seřadí od největších výskytů
  - head -10 zobrazí prvních 10 řádků

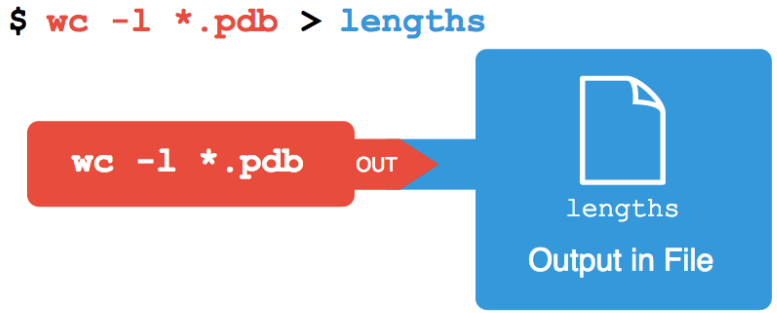
Roury – příklad – 10 nejvyskytovanějších slov

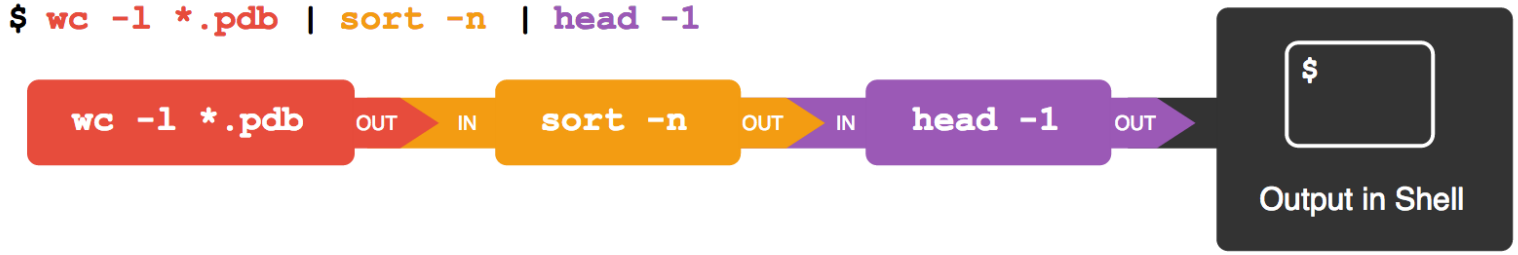
```
root@mesosdev:/home/mesosdev-api/htdocs/api.mesosdev.cz/mesosAPI# cat main.py |
tr ' ' '\n' | sort | uniq -c | sort -nr | head -10
   3026
     86 =
     52 return
     44 """
     36 def
     31 if
     25 pro
     21 ==
     21 "",
     18 :return:
```


MěSOŠ Klobouky u Brna

Proč použít roury







MěSOŠ Klobouky u Brna

Pojmenování rour

- Pro **komunikaci mezi procesy nepřímo propoj**ené
- Také se nazývají jako **FIFO** (First In First Out)
- Vytváříme pomocí **mkfifo nazevRoury**
- Způsob meziprocesové komunikace (IPC)
  - proces 1 zapíše do roury (logy, aplikace)
  - proces 2 čte data z roury (monitorovací nástroj)
- Jedná se o **buffer** mezi zapisovatelem a čtenářem

FILTRY

Filtry

- Speciální programy v Linuxu
- Zpracovávají textová data – např. vstup z stdin
- **Generují upravený výstup**
- Často se kombinují s rourama

cat

- **Čte** data z jednoho souboru
- Případně **spojí data** z více souborů do jednoho
- cat /etc/passwd
  - vypíše obsah souboru /etc/passwd

grep

- **Vyhledává řetězce nebo vzory v textu**
- Přepínače
  - **-i** ignoruje velikost znaků
  - **-v** negace – zobrazí řádky neobsahující výraz
  - **-r** rekurzivně vyhledává v adresářích

```
grep "chyba" log.txt #vyhledá řetězec v souboru
grep -i "chyba" log.txt #hledá bez ohledu na velikost znaků
grep -v "chyba" log.txt #vyhledá řádky neobsahující řetězec "chyba"
```

sort

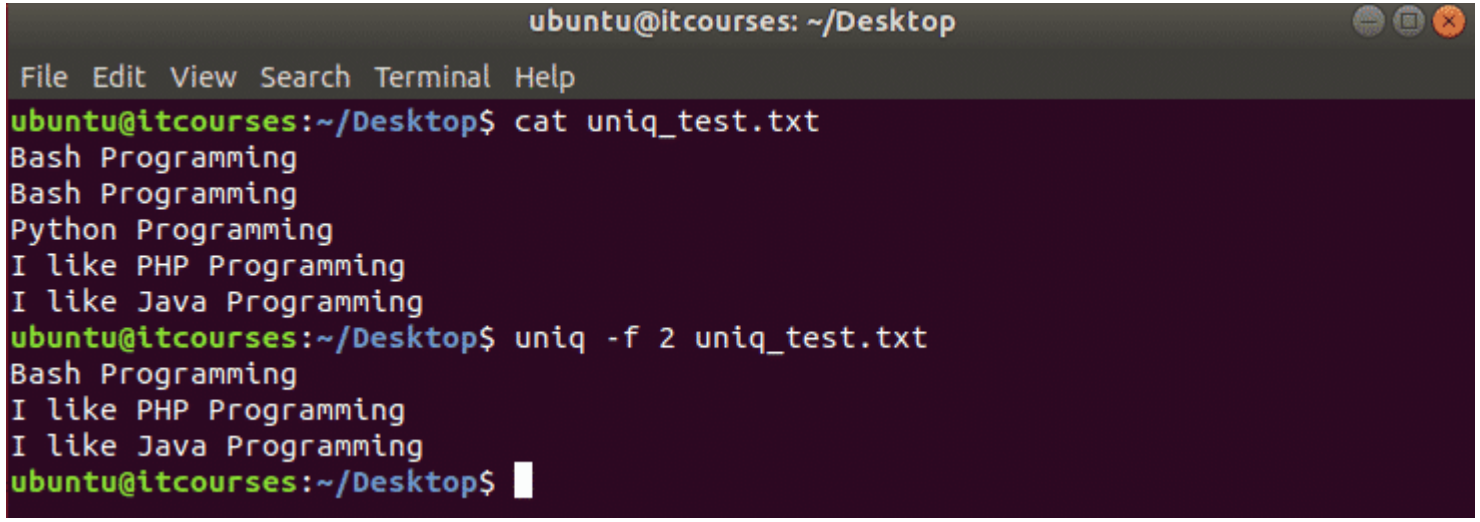
- **Setřídí obsah** podle řádků
  - standardně podle abecedy A-Z
- Přepínače
  - **-r** třídí obráceně (sestupně)
  - **-n** třídí podle čísel
  - **-k** třídí podle sloupce

sort

```
    "Adam Černý"
    "Adam Noé Mančuška"
#!/bin/bash
    clpctl site:add:php \
    "Denis Vrba"
    domain="${login}.mesosdev.cz"
        --domainName="$domain" \
done
    echo "${clean_last_name}${clean_first_initial}"
echo "Hotovo!"
    echo "Vytvářím stránku pro: $student"
for student in "${students[@]}"; do
generate_login() {
    "Jakub Dočkal"
    "Jan Píza"
    "Kryštof Nečas"
```

uniq

- Odstraní **duplikátní řádky**
  - které jdou za sebou
- Přepínače:
  - **-c** počítá výskyt každého řádku
  - **-d** zobrazí pouze duplikátní hodnoty



head a tail

- **head** – zobrazí **první řádky** ze souboru
- **tail** – zobrazí **poslední řádky** ze souboru
- Přepínače:
  - **-n X** upřesňuje, kolik zobrazí řádku (**10 výchozí**)
  - **-f** sleduje soubor v reálném čase – logy, v komb. s tail)

```
lukino185@mesosdev:~/sh-skripty$ head -n 4 I4_24_25_stranky.sh
#!/bin/bash

students=(
    "Silvio Bellino"
```

wc

- **Počítá** řádky, slova nebo znaky
- Přepínače:
  - **-l** počítá řádky
  - **-w** počítá počet slov
  - **-c** počítá počet znaků

```
lukino185@mesosdev:~/sh-skripty$ wc I4_24_25_stranky.sh
  55  127 1404 I4_24_25_stranky.sh
lukino185@mesosdev:~/sh-skripty$ wc -l I4_24_25_stranky.sh
55 I4_24_25_stranky.sh
lukino185@mesosdev:~/sh-skripty$ wc -w I4_24_25_stranky.sh
127 I4_24_25_stranky.sh
```

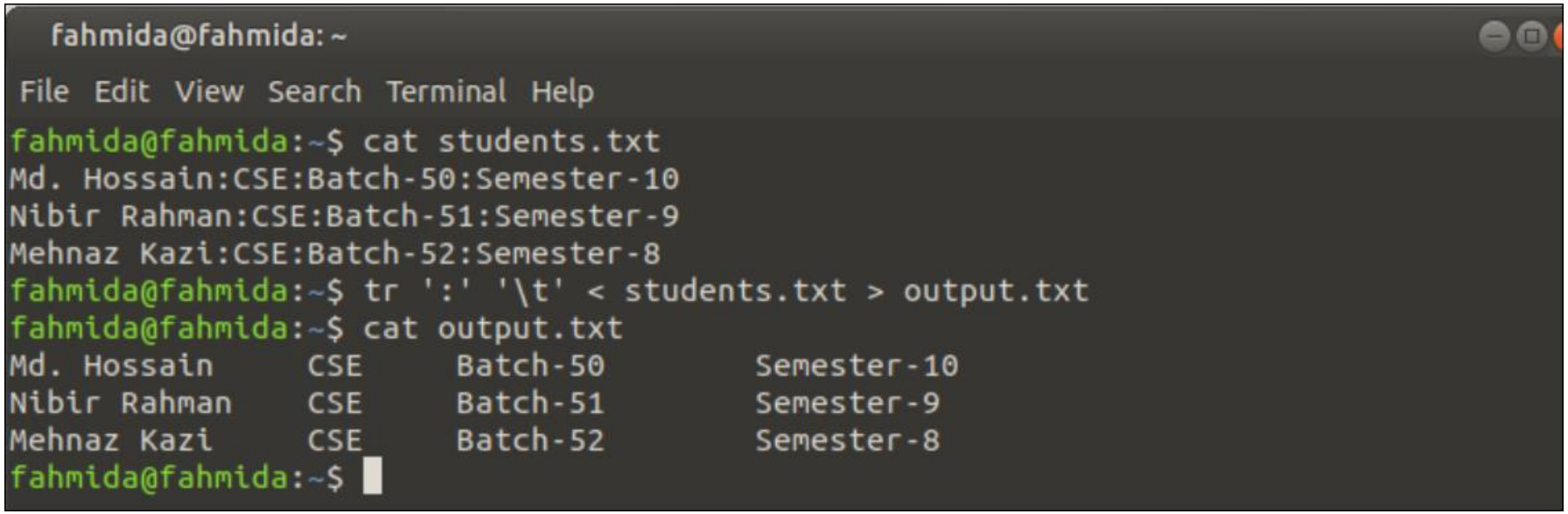
cut

- **Extrahuje** sloupce nebo části řádků
- Přepínače
  - **-d X** určuje oddělovač (čárka, středník, mezera…)
  - **-f X** vybírá sloupce dle pořadí

```
lukino185@mesosdev:~/sh-skripty$ cut /etc/passwd -d ':' -f 4,5,6
0:root:/root
1:daemon:/usr/sbin
2:bin:/bin
3:sys:/dev
65534:sync:/bin
60:games:/usr/games
```

tr

- **Převádí nebo odstraňuje znaky**
- Přepínače:
  - **-d** odstraní zadané znaky
  - **-s** sloučí opakující se znaky



SKRIPTY

Skripty

- Podobný princip a užití jako .bat u Windows
- Nemusíme psát dokola stejné příkazy
- V Linuxu – **shellové skripty**
  - přípona **.sh**
- Skript je **sekvence příkazů napsané v txt souboru, které systém provede jako celek**

Vytvoření skriptu

- Vytvoříme pomocí **nano nazevSkriptu.sh**
- Na první řádek uvedeme **interpret** (**Shebang**)
  - **#!/bin/bash**
- Následují již samotné příkazy
- Nakonec skript označíme jako **spustitelný**
  - **chmod +x nazevSkriptu.sh**
- Skript spustíme příkazem
  - **./nazevSkriptu.sh**

Konstrukce

- **Operátory**
  - **-eq** rovnost (equal)
  - **-ne** nerovnost (not equal)
  - **-lt** menší než (less than)
  - **-le** menší nebo rovno (less than or equal to)
  - **-gt** větší než (greater then)
  - **-ge** větší nebo rovno (greater than or equal to)

Konstrukce

- **Proměnné**

  jmeno = „Lukáš“

  „Ahoj, $jmeno“

- **Podmínky if**

```
if [  $vek –ge 18 ]; then
        echo „Jsi plnoletý“
else
        echo „jsi nezletilý“
fi
```

Konstrukce

- **For cykly**

```
for soubor in *.txt do
    echo „zpracovávám soubor“ $soubor
done
```

- **While cykly**

```
pocet = 0
while [ $pocet -lt 5 ]; do
    echo „Počet je: $pocet“
    pocet = $((pocet + 1))
done
```

Konstrukce

- **Vstup od uživatele**

```
echo „Zadej jmeno“
read jmeno
echo „Ahoj, $jmeno“
```

- **Funkce**

```
pozdrav() {
        echo "Ahoj, $1!
}
pozdrav "Lukáš"
```

Konstrukce

- **Argumenty**

  echo "První argument: $1"

  echo "Druhý argument: $2„

- **A následné volání s argumenty**

  ./muj_skript.sh argument1 argument2

PLÁNOVÁNÍ ÚLOH

Cron

- Pro **pravidelné spuštění úloh**
- Systémový démon umožňující spouštět úlohy v nastavených intervalech
- Používá se u zálohování, webových skriptech atd…

MěSOŠ Klobouky u Brna

Crontab

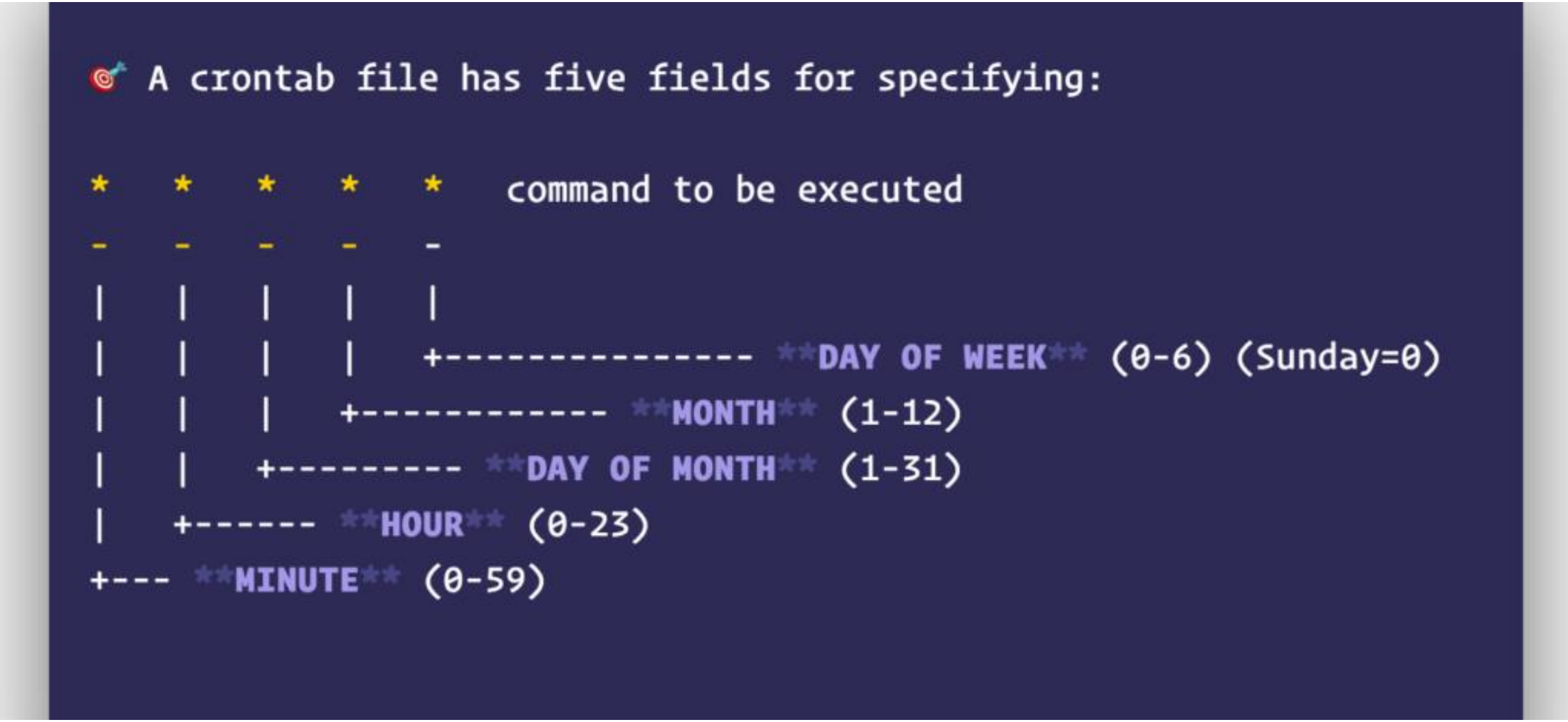
- Pomocí crontab provádíme **konfiguraci**
  - definujeme, kdy a co se má spustit
- Rozdělení
  - **uživatelský**
    - /var/spool/cron/crontabs/<uživatel>
  - **globální**
    - /etc/crontab
    - /etc/cron.d

Formát crontabu

- Jednotlivé řádky obsahují:
  - **5 polí času**
  - 6. pole – **jméno uživatele** (pouze u systémového)
  - **spuštěný příkaz**
- Hvězdička – v určitém časovém bloku spouští **vždy**

```
*/5 * * * * /usr/bin/php8.4 /home/mesosdev-moodle/htdocs/moodle.mesosdev.cz/admin/cli/cron.php
```

Formát crontabu



Příklady

- Každý pracovní den v 6:00
  - 0 6 * * 1-5 /home/user/backup.sh
- Každou minutu
  - * * * * * echo "Minutka"
- Každých 5 minut
  - */5 * * * * wget mesos.cz
- Každých 10 minut od 9:00 do 17:50 každý pracovní den, pod uživatelem root
  - */10 9-17 * * 1-5 root /home/pospisill/skript.sh

Správa crontab

- Zobrazení crontab aktuálního uživatele
  - **crontab -l**
- Úprava crontab
  - **crontab -e**
- Odstranění všech naplánovaných úloh
  - **crontab -r**

at

- Pro **jednorázové spuštění úlohy**
- Říká, za jak dlouho se příkaz provede
- Využíváme jednoduché časové výrazy

MěSOŠ Klobouky u Brna

Příklady

- Dnes ve 14:30
  - echo "ls > vypis.txt" | at 14:30
- Dne 1. 6. 2025 ve 14:00
  - wget mesos.cz | at 14:00 01/06/25

Další příkazy

- **Zobrazení** naplánovaných úloh
  - **atq**
- **Zrušení** úlohy
  - **atrm <číslo_úlohy>**

```
root@kMeghana:~# atq
16      Tue Feb 26 16:00:00 2019 m root
15      Tue Feb 26 02:15:00 2019 m kum
17      Sun Mar  3 02:15:00 2019 m kum
13      Tue Feb 26 07:50:00 2019 a kum
12      Mon Mar  4 18:30:00 2019 a kum
14      Fri Mar  1 20:30:00 2019 m kum
18      Tue Feb 26 07:00:00 2019 a root
19      Tue Feb 26 06:35:00 2019 a root
```

Anotace

Tato prezentace slouží k seznámení se s principy pokročilé práce s terminálem v OS Linux. Obsahuje přehled práce s datovými proudy, přesměrováním, rourami a jejich využitím, včetně základních filtrů pro manipulaci s textovými daty. Dále se žáci naučí vytvářet jednoduché shellové skripty pro automatizaci běžných úloh a principy plánování úloh.

Hlavním cílem je, aby si žáci osvojili základní i pokročilé techniky práce s terminálem a uměli je aplikovat při správě systému.

Užité zdroje

- https://vyuka.mesosdev.cz/kb/operacni-systemy/linux/pokrocila-prace-s-terminalem/

Autor: Bc. Lukáš Pospíšil

Předmět: Operační systémy

Ročník: 4.

Rok vytvoření: 2024

Poslední aktualizace: 2024

MěSOŠ Klobouky u Brna