Bash III

Eliška Jégrová

2. 11. 2025

Obsah

Proměnné

- Příklad přiřazení

```
$ jmeno=minotaur.dat
```

- Obsah proměnné

```
$ echo $jmeno
```

- Proměnnou lze použít v příkazu, např.:

```
$ head -n2 $jmeno | tail -n1
```

- Hodnota proměnné můžeme přepsat

```
$ jmeno=unicorn.dat
$ head -n2 $jmeno | tail -n1
```

Cyklus `for`

- Syntaxe:

```
for prvek in seznam
  do
   prikaz
done
```

- Příklad s proměnnou:

```
for jmeno in minotaur.dat unicorn.dat
  do
   head -n1 $jmeno
done
```

- Příklad - vypsání souborů:

```
for soubor in *.dat; do
  echo $soubor
done
```

Cyklus `for` – vnořené cykly

- Příklad: Vytvořit složky z názvů molekul a určitých teplot

```
for molekula in cubane ethane methane
do
    for teplota in 25 30 37 40
    do
        echo mkdir $molekula-$teplota
    done
done
```

Samostatná práce 1 – Proměnné a cykly

- **Cíl:** procvičit práci s proměnnými a cyklem `for` při tvorbě více složek.
- Vytvoř proměnnou `projekty` s názvy projektů "web api data"
- Pomocí cyklu vytvoř adresáře s názvy z proměnné.
- Zkontroluj vytvoření pomocí `ls`.
- Bonus: Pomocí cyklu vytvoř v každém projektu podadresář `logs`, aby struktura vypadala takto:

```
.
├── api
│   └── logs
├── data
│   └── logs
└── web
    └── logs
```

- Bonus 2: přidej do každého podadresáře soubor `info.txt` s textem "Projekt <jméno>".

Proměnné prostředí

- Proměnné bashe - jsou dostupné jen pro samotný Bash
- Proměnné prostředí - dostupné i pro běžící programy

- Příkaz `env` vypíše všechny proměnné prostředí:

```
$ env | head
$ env | grep ^LANG
```

Exportování

- způsob, jak předat proměnnou programům

```
$ export jmeno=basilisk.dat
$ export jmeno
```

- Příklad

Bez exportu:

```
jmeno=minotaur.dat
env | grep jmeno
echo $jmeno
minotaur.dat
```

S exportem:

```
export jmeno
env | grep jmeno
jmeno=minotaur.dat
echo $jmeno
minotaur.dat
```

Složení názvů proměnných, Výzva

- Pokud chceš mít text za proměnnou (např. „ovoce“ v „jablko“), použij složené závorky:

```
$ majitel=Petr
$ echo ${majitel}ovo jablko
```

- Bash má vlastní proměnné používané pro prompt: `PS1`, `PS2`.
  - Příklad přenastavení výzvy:

```
$ PS1='[0_0]'
```

Samostatná práce 2 – Proměnné prostředí a výzva

- **Cíl:** vyzkoušet práci s proměnnými prostředí a úpravou výzvy.
- Nastav proměnnou `jmeno` s tvým jménem a exportuj ji.
- Ověř, že je dostupná v `env`
- Uprav výzvu (`PS1`), aby ukazovala tvoje jméno
- Bonus: Přidej k výzvě i aktuální adresář `\w`.

Skripty

- Skripty jsou jednoduché programy – často spojují existující příkazy.
- Ukázka obsahu `klasifikace`:

```
for x in *.dat
do
  head -n2 $x | tail -n1
done
```

Spustitelný skript

- Skript se spouští pomocí Bashe:

```
$ bash klasifikace
```

- Aby šel spustit jako příkaz, je třeba:
  1. Přidat **shebang** na první řádek:

```
#! /bin/bash
```

  2. Nastavit spustitelný příznak:

```
$ chmod +x klasifikace
```

  3. Spustit:

```
$ ./klasifikace
```

Argumenty skriptu

```
$ ./klasifikuj minotaur.dat
```

- Argumenty z příkazové řádky se předávají pomocí speciálních proměnných
  - `"$0"` – jméno skriptu
  - `"$1"` – první argument
  - `"$2"`, `"$3"` – další argumenty
  - `"$@"` – všechny argumenty

- Použití ve skriptu:

```
#! /bin/bash
head -n2 "$1" | tail -n1
```

Proměnná PATH

- Systém hledá spustitelné programy v adresářích uvedených v proměnné `PATH`.

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

- Skript přesuneme do složky v `PATH`, např. `$HOME/bin`
- Spustíme skript jen názvem:

```
$ klasifikace *.dat
```

- K trvalé změně `PATH` je potřeba upravit ji v souboru `.bashrc`

Další kombinace příkazů

- Příkazy lze řetězit pomocí ; – provedou se postupně:

```
$ echo "Seznam souboru:"; ls
```

- Operátor && – druhý příkaz se provede jen pokud první uspěl:

```
$ mkdir test && echo "Slozka vytvorena."
```

- Operátor || – druhý příkaz se provede jen pokud první selhal:

```
$ cat neexistujici.txt || echo "Soubor nenalezen."
```

Samostatná práce 3 – Vytvoření skriptu

- **Cíl:** vytvořit jednoduchý skript, který vypíše základní informace o systému.
- Vytvoř soubor `info` s obsahem:

```
echo "Uzivatel: $USER"
echo "Aktualni adresar: $(pwd)"
echo "Datum: $(date)"
```

- Nastav skript jako spustitelný
- Přesuň ho do adresáře `$HOME/bin` (vytvoř jej, pokud neexistuje).
- Spusť ho odkudkoliv jen názvem:

```
$ info
```

Jak Bash hledá příkazy

- Příkaz `type` ukáže, k jakému typu příkaz patří
- Když do Bashe zadáš jméno příkazu (např. `ls`), Bash hledá v několika druzích příkazů:
  - programy v adresářích z `$PATH`

```
$ type cat
cat je /usr/bin/cat
```

  - zabudované příkazy (built-in)

```
$ type cd
cd je soucast shellu
```

  - aliasy – zkratky pro jiné příkazy

```
$ type ll
ll je alias na "ls -l --color=auto"
```

  - funkce shellu – skripty přímo definované v Bashi

```
$ type quote
quote je funkce
```

Skript pro vytvoření a vstup do složky

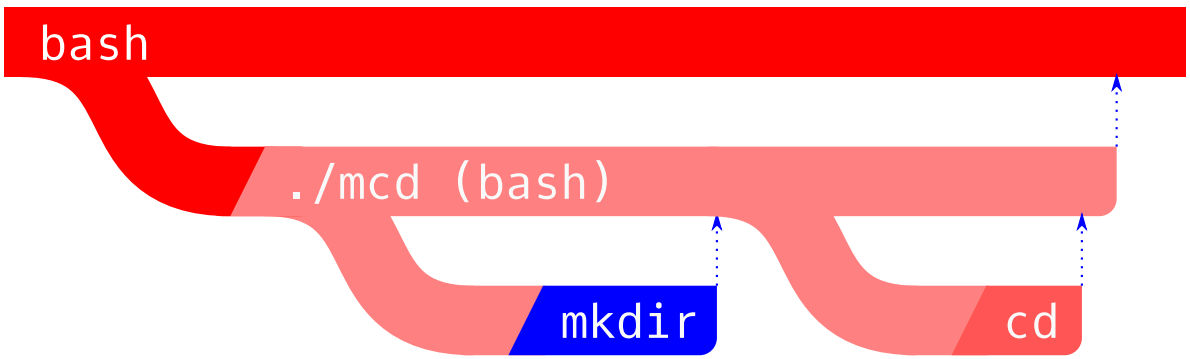
- Cílem - skript, který vytvoří novou složku a přepne se do ní.
- Obsah skriptu `mcd`:

```
mkdir -p $1
cd $1
```

- Spuštění:

```
$ bash mcd novy-adresar
```

- Nový proces spouští skript – změna adresáře se ale týká jen tohoto podprocesu.



Zabudované příkazy a `source`

- Některé příkazy jsou součástí samotného shellu – tzv. **zabudované (built-in)** příkazy.
  - Např. `cd`, `export`, `alias`, `source`
  - Nespouštějí nový proces – mění přímo prostředí aktuálního shellu

- Ověření, zda je příkaz zabudovaný:

```
type cd
# cd is a shell builtin
```

- Příkaz `source` (nebo zkráceně `.`) spustí skript v aktuálním shellu.

```
source mcd novy-adresar
```

- Díky tomu zůstaneš v nové složce i po skončení skriptu – na rozdíl od běžného spuštění, které vytváří nový proces.

Alias – jednoduchá zkratka

- Alias je zkratka - nahrazuje jeden příkaz jiným

```
alias gs='git status'
```

- Použití:

  když napíšeš `ll /home`, Bash místo toho spustí `ls -l -color=auto /home`.

- Zrušení aliasu:

```
unalias gs
```

Funkce v Bashi

- Funkce je více „robustní“ zkratka – může obsahovat více příkazů, logiku, podmínky.
- Definice:

```
mcd() {
        mkdir -p "$1"
        cd "$1"
        }
```

- Použití:

```
mcd projekty
```

- Argumenty uvnitř: `$1`, `$2`, ..., počet v `$#`, všechny v `$@`.
- Užitečný příklad – kombinace mkdir + cd:

```
mcd() {
  mkdir -p "$1" && cd "$1"
}
```

Zapamatování aliasů & funkcí (např. v `.bashrc`)

- Alias nebo funkci definujeme ručně, ale zmizí po ukončení Bashe.
- Řešení: vložit je do souboru `~/.bashrc`, který se načte při startu shellu:

```
alias gs='git status'
mcd() {
        mkdir -p "$1" && cd "$1"
}
```

- Po úpravě spustíme:

```
$ source ~/.bashrc
```

- Nebo otevřít nový terminál — definice se načtou.

Samostatná práce 4 – Typy příkazů, aliasy a funkce

- **Cíl:** pochopit rozdíl mezi typy příkazů a vytvořit vlastní alias a funkci.
- Zjisti typ příkazů `ls`, `source`, `alias`
- Vytvoř alias pro často používaný příkaz, např. něco z gitu.
- Definuj jednoduchou funkci `greet`, která bude obsahovat

```
echo "Ahoj $USER!"
```

- Spusť:

```
$ greet
```

- Bonus: přidej alias i funkci do `~/.bashrc` a načti je v terminálu (source)