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

PSK3-9

Název školy:	Vyšší odborná škola a Střední průmyslová škola, Božetěchova 3
Autor:	Ing. Marek Nožka
Anotace:	Základy skriptování v unixovém shellu
Vzdělávací oblast:	Informační a komunikační technologie
Předmět:	Počítačové sítě a komunikační technika (PSK)
Tematická oblast:	Operační systém Linux/Unix
Výsledky vzdělávání:	Žák vytvoří jednoduchý skript pro příkazové prostředí
Klíčová slova:	Linux, Unix, shell, bash, skript
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

Základy skriptování

Unixový shell se dá použít nejen pro interaktivní práci, ale i jako programovací jazyk. Jednotlivé příkazy můžeme zapsat do souboru a tento soubor nechat interpretovat příkazovým interpretem. Tento soubor je potom označován jako *skript* nebo shellový skript.

Hlavička

Aby bylo možné *skript* jednoduše spouštět je nutné opatřit ho hlavičkou: První řádek začíná dvojicí znaků #! a pokračuje cestou k příkazovému interpretu:

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

`--> stáhnout

Tento mechanizmus je zcela obecný a uplatní se stejně i pro skripty v jiný jazycích. Například pro programovací jazyk Python, by hlavička mohla vypadat takto:

```
#!/usr/bin/python
```

`--> stáhnout

Aby se skript choval jako každý jiný program je jeho správná hlavička podmínkou nutnou nikoli postačující. Skript musí být spustitelný1 a musí být umístěn tak, aby ho příkazový interpret našel. Pokud je vše toto splněno může ho uživatel spouštět, jako by to byla běžná součást sytému.

```
$ mujSkript.sh
```

Příkaz . (tečka)

Ano . je opravdu příkaz. Výše popsaný způsob spouštění má za následek, že námi vytvořený skript je spuštěn v *podřízeném shell-u*. To mimo jiné znamená, že změny provedené v proměnných prostředí nebudou po dokončení skriptu viditelné. Použijeme-li příkaz:

```
$ . soubor
```

...bude shell postupně číst `soubor` a jednotlivé příkazy vykonávat v aktuálním shellu. Výsledek bude stejný jako kdyby uživatel zadával příkazy rovnou v interaktivním prostředí.

Poznámky

Poznámky se zapisují za znak `#`. Vše od znaku `#` do konce řádku je interpretem ignorováno.

Přebírání parametrů (argumentů)

Skriptům lze stejně jako programům předávat parametry. Ty jsou uvnitř skriptu dostupné po mocí speciálních proměnných.

`$0`	jméno skriptu
`$1`	první parametr
`$2`	druhý parametr
`$`*n*	*n*-tý parametr, *n* nabývá hodnoty 1 až 9
`${`*n*`}`	libovolný *n*-tý parametr
`$#`	číslo posledního parametru
`$*`	seznam všech parametrů -- stejné jako `"$1 $2 $3 ..."`
`$@`	seznam všech parametrů -- stejné jako `"$1" "$2" "$3" ...`

Příkaz `shift` *N* vymaže prvních *N* parametrů a posune význam proměnných `$`*n* a `${`*n*`}`

```
1 #!/bin/bash
2 # Soubor:    mujSkript.sh
3 #  Popis:    Ukázka předávání parametrů
```

```
 4 ############################################################
 5
 6 echo Jmenuji se: $0
 7 echo Parametry: $@
 8 echo první: $1
 9 echo třetí: ${3}
10
11 shift 3
12 echo ">>> první 3 parametry byli vymazány"
13 echo Parametry: $@
14
```

`--> stáhnout

```
$ ./mujSkript.sh toto je ukazka predavani parametru
Jmenuji se: ./mujSkript.sh
Parametry: toto je ukazka predavani parametru
první: toto
třetí: ukazka
>>> první 3 parametry byli vymazány
Parametry: predavani parametru
```

Návratová hodnota

Jak bylo uvedeno2 každý program, který ukončí svou činnost korektně vrací jako svou návratovou hodnotu `0`. Pokud se v průběhu programu objeví chyba, dává o tom program vědět svou nenulovou návratovou hodnotou.

Okamžité ukončení skriptu s konkrétní návratovou hodnotou vyvolá příkaz `exit` *N*.

```
 1 #!/bin/bash
 2 # Soubor:    navrat.sh
 3 #  Popis:    Ukázka návratové hodnoty
 4 ############################################################
 5
 6 if [ $1 ] && [ $1 = 'ano' ]; then
 7     echo OK
 8     exit 0
 9 else
10     echo napiš ano
11     exit 1
12 fi
13
```

`--> stáhnout

```
$./navrat
napiš ano
$ echo $?
1

$./navrat NEE
napiš ano
$ echo $?
1

$./navrat ano
OK
$ echo $?
0
```

Funkce

Shell umožňuje vytvářet funkce. Funkce se chová stejně jako samostatný skript, včetně předávání parametrů a návratové hodnoty.

Ukončení funkce s konkrétní návratové hodnotou je zajištěna příkazem `return N`.

Sada proměnných pro předávání parametrů funguje stejně jako u skriptu.

Syntaxe funkce v shellu:

```
název() {
    příkaz
    příkaz
    ....
}
```

```
 1 #!/bin/bash
 2 # Soubor:    scitani.sh
 3 #  Popis:    Ukázka funkce
 4 ###############################################################
 5
 6
 7 plus() {
 8     echo ${1}+${2}=$[ $1 + $2 ]
 9 }
10
11 plus $1 $2
12 plus $3 $4
13 plus $5 $6
```

`--> stáhnout

```
$ ./scitani.sh 1 2 3 4 5 6
1+2=3
3+4=7
5+6=11
```

Podmínky cykly

Užití podmínek a cyklů není vázáno jen na skripty. Stejně tak je možné je použít v interaktivní práci.

V následujících ukázkách jsem zvolil způsob zápisu, který mi připadá přehledný, ale platí, že středník ; může být zaměněn za konec řádku a naopak.

if -- podmíněné vykonání

Syntaxe obecně vypadá takto:

```
if VYRAZ; then
    PRIKAZ
```

```
    ....
fi
```

`--> stáhnout

```
if VYRAZ; then
    PRIKAZ
    ....
else
    PRIKAZ
    ....
fi
```

`--> stáhnout

```
if VYRAZ; then
    PRIKAZ
    ....
elif VYRAZ; then
    PRIKAZ
    ....
elif VYRAZ; then
    PRIKAZ
    ....
...
else
    PRIKAZ
    ....
fi
```

`--> stáhnout

VYRAZ:

- VYRAZ je příkaz nebo posloupnost příkazů oddělených pomocí metaznaku `||` (logický součet -- nebo) nebo metaznaku `&&` (logický součin -- and).
- O pravdivosti nebo nepravdivosti výrazu rozhoduje jeho **návratová hodnota**.
- VYRAZ je možné negovat pomocí znaku `!` na jeho začátku.

Jako `VYRAZ` se velice často používá **program `test`** respektive jeho synonymum **`[`**. Podmínka potom vypadá **jako** by byla zapsána do hranatých závorek, ale ve skutečnost je volán program `test`.

Program test:

1. porovnává řetězce
2. porovnává celá čísla
3. testuje typy souborů a jejich stáří

Více se dočtete v manuálové stránce programu test.

```
1 #!/bin/bash
2 # Soubor:    if.sh
3 #  Popis:    Ukázka podmínky
4 ###############################################################
5
6 if [ -z $1 ]; then
7     echo program nemá žádné parametry
8 else
9     echo program má $# parametrů
```

```
10 fi
11
```

`--> stáhnout

case -- shoda se vzorem

Vícenásobné větvení pomocí vzorů ukážeme na příkladu:

```
 1 #!/bin/bash
 2 # Soubor:     case.sh
 3 #  Popis:     Ukázka CASE
 4 ###############################################################
 5
 6 case $1 in
 7     ahoj|nazdar|cau) echo pozdrav;;
 8     [tT]*) echo slovo zacina na T;;
 9     *e*) echo slovo obsahuje e;;
10     *) echo cokoliv jiného;;
11 esac
```

`--> stáhnout

- Větev ukončují dva středníky.
- Znak `|` slouží jako oddělovač vzorů ve významu *nebo*.
- Pro vzory platí stejná pravidla jako pro expanzi jmen souborů.

for -- pro každou položku seznamu

Syntaxe cyklu `for` vypadá takto:

```
for PROMENNA in SEZNAM; do
    PRIKAZ
    PRIKAZ
    .....
done
```

`--> stáhnout

Nejprve se expanduje SEZNAM. PROMENNA nabývá při každé iteraci postupně jedné z hodnot SEZNAMu.

```
for cislo in 1 2 3 4; do
    echo cislo je $cislo
done
```

`--> stáhnout

Jako seznam může ale být uvedeno neúplné jméno souboru

```
for soubor in *; do
    echo menim casove razitko: $soubor
    touch $soubor
done
```

`--> stáhnout

... nebo program, který seznam vrátí.

```
for cislo in $(seq 5 10); do
    echo cislo je $cislo
done
```

`--> stáhnout

while, until -- opakování na základě podmínky

Syntaxe cyklu `while` a `until` vypadá naprosto stejně. Jediný rozdíl je v podmínce. Iterace cyklu `while` se vykoná pokud podmínka platí. Iterace cyklu `until` se vykoná pokud podmínka neplatí.

K okamžitému ukončení těla cyklu slouží příkaz `break` k přeskočení zbytku těla cyklu příkaz `continue`.

```
while VYRAZ; do
    PRIKAZ
    PRIKAZ
    ....
done
```

`--> stáhnout

VYRAZ má stejný význam jako u podmínky `if`.

```
while ping -c 3 172.16.6.53 &>/dev/null; do
    echo PC je zapnute.
done
echo K PC se není možné připojit.
```

`--> stáhnout

Standardní vstup s těla skriptu

Častokrát požadujeme, aby program spuštěný z těla skriptu byl "nakrmen" vstupními daty. Pro přesměrování standardního vstupu ze souboru sice slouží metaznak `<` ale vytváření samostatného souboru pro vstupní data není vždy efektivní. Proto použijeme metaznak `<<`.

```
prikaz <<OMEZOVAC
DATA
DATA
DATA
....
OMEZOVAC
```

`--> stáhnout

- OMEZOVAC je libovolná posloupnost znaků.
- Ukončovací OMEZOVAC musí být uveden nasamostatném řádku.
- Ve vstupních datech je možné zapsat i proměnné.
- pokud je "OMEZOVAC" v uvozovkách dereference proměnných se neprovádí.

```
cat <<EOF | wc -w
u textu, tery jsem sem napsal se provede
pocitani slov. Pocet slov je:
EOF
```

`--> stáhnout

---

1. `chmod a+x ~/bin/mujSkript.sh` ↩
2. v kapitole Návratová hodnota a Oddělování příkazů ↩