

Cvičení 4.

ZOS 2013



Pravdivostní hodnota příkazu

- proces po skončení předává **návratovou hodnotu** (exit status)
- konvence
 - **0** – úspěšné ukončení příkazu
 - **jiné** – neúspěch
 - **128+n** - ukončení signálem (n je číslo signálu)

Při neúspěchu nás zajímá důvod (různé návratové kódy),
při úspěchu stačí status OK, vyzkoušejte: **echo \$?**



Potomek procesu

- potomek procesu zdědí **kopii prostředí** svého rodiče (proměnné, akt. adresář)
- potomek může prostředí měnit, ale změny se **nedotknou** rodiče – mění **kopii** původních dat

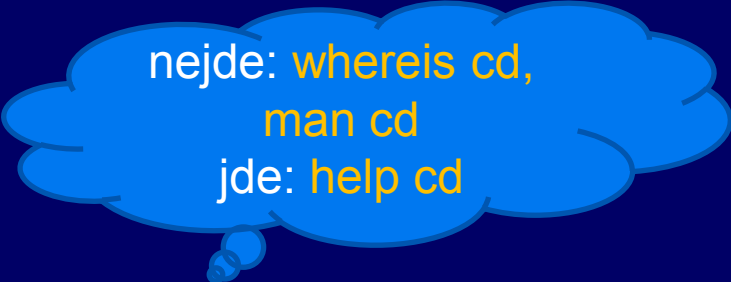
Př.

1. Vypíšeme aktuální adresář (pwd)
2. Spustíme skript, který změní adresář (cd /)
3. Po skončení skriptu je akt. adresář nezměněn

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



Příkazy



nejde: `whereis cd`,
`man cd`
jde: `help cd`

- **vestavěné** příkazy shellu (`cd`, `set`)
- **externí** příkazy (`ls`, `cp`, `mv`)
 - spustitelné soubory (v `/bin`, `/sbin`, `/usr/bin`,...)
 - shell je spustí jako svého potomka
- **příkazový soubor** (shell script)
 - očekává textový soubor obsahující příkazy shellu
 - shell spustí svojí kopii a ta provede příkazy



Příkazový soubor (skript)

- shell zkoumá první řádek
- pokud obsahuje **#! řetězec**
 - **řetězec** – jméno programu, kterým bude soubor interpretován (bash, jiný shell, perl, ...)
 - systém tento program spustí a předá mu jméno souboru jako argument

#! /bin/bash

echo Jsem skript pusteny bashem



Příklad – zdravící skript

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

neodřádkuje

```
echo -n "Zadej své jméno: "
```

```
read PREZDIVKA
```

```
echo Nazdarek, ty jsi $PREZDIVKA
```

```
echo Prihlaseny jako $USER
```

systemová proměnná

□ Vyzkoušejte a odpovězte na otázky na dalším slidu



Otázky k předchozímu skriptu

Po spuštění skriptu zadejte příkaz **set**.
Uvidíte proměnnou PREZDIVKA? Zdůvodněte!

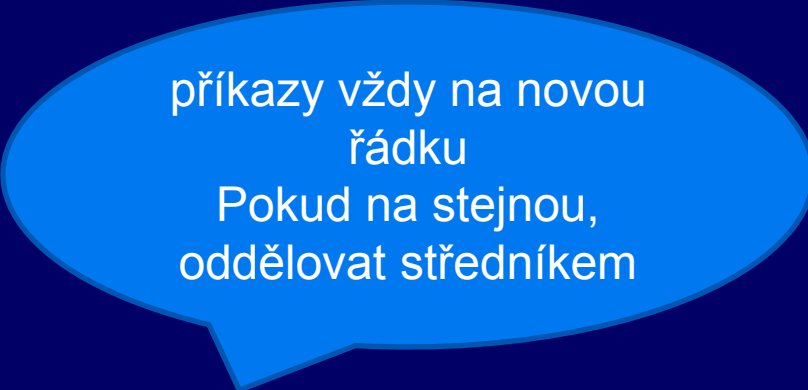
Jaký je rozdíl v provedení následujících příkazů?

1. `zdrav_skript`
2. `zdrav_skript Lada`
3. `zdrav_skript < vstup.txt` (vstup.txt obsahuje: Tomas)



Podmíněné příkazy - if

```
if seznam-prikazu
then seznam-prikazu
[ elif seznam-prikazu
  then seznam-prikazu ] ...
[ else seznam-prikazu ]
fi
```



příkazy vždy na novou
řádku
Pokud na stejnou,
oddělovat středníkem

- vykoná se seznam příkazů za if, pokud návratová hodnota 0, provedou se příkazy za then
- if seznam-prikazu ; then seznam-prikazu ; fi



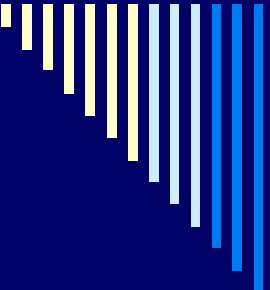
Příklad - if

```
#!/bin/bash
# skript testuje, co je pivo
if test -d pivo
then
    echo "pivo je adresar"
elif test -f pivo
then
    echo "pivo je soubor"
else
    echo "pivo neni adresar ani soubor"
fi
```

test -d file
test na adresář

test -f file
test na obyčejný
soubor

Lze zapsat:
[-f file]
důležité jsou mezery
kolem závorek



Příklad if2 – test pingu

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

```
echo "zadej jmeno stroje: "  
read STROJ  
if ping -q -c 3 $STROJ > /dev/null  
then  
    echo "Stroj $STROJ pinguje"  
else  
    echo "Stroj $STROJ nepinguje"  
fi
```

Praktický
příklad,

Využívá
návratový kód
příkazu ping



Podmíněný příkaz - case

case slovo **in**

vzor) seznam-prikazu ;;

vzor) seznam-prikazu ;;

...

esac

- srovnává slovo se vzorem
- pokud souhlasí, vykoná seznam příkazů a skončí



Příklad – case, pípnutí, systémová proměnná

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

```
# skript zpracuje první parametr
```

```
case $1 in
```

```
-h | -help) echo " Napoveda: spust me s jednim parametrem"
```

```
    echo " -c, -d, -p nebo -b"
```

```
    ;;
```

```
-c)    echo " Vypisi kalendar" ; cal ;;
```

```
-d)    echo " Dnes mame:" ; date ;;
```

```
-p)    echo -e " Ted pipnu... \a " ;;
```

```
-b)    echo " $USER je borec " ;;
```

```
*)    echo " Spust me s parametrem -h pro napovedu "
```

```
    ;;
```

```
esac
```



Příkaz cyklu - for

for proměnná **in** seznam-slov

do

seznam-příkazů

done

- ❑ nejdříve expandován seznam slov
- ❑ oddělena mezerou



Příklad – for1

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

```
# zkopíruje všechny soubory z aktuálního adresáře
```

```
# s příponou .kuk na .kuk.backup
```

```
for F in *.kuk
```

```
do
```

```
    cp $F ${F}.backup
```

```
done
```



Příklad – for2

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

```
for den in patek sobota nedele  
do  
    echo Oblíbený den je $den  
done
```

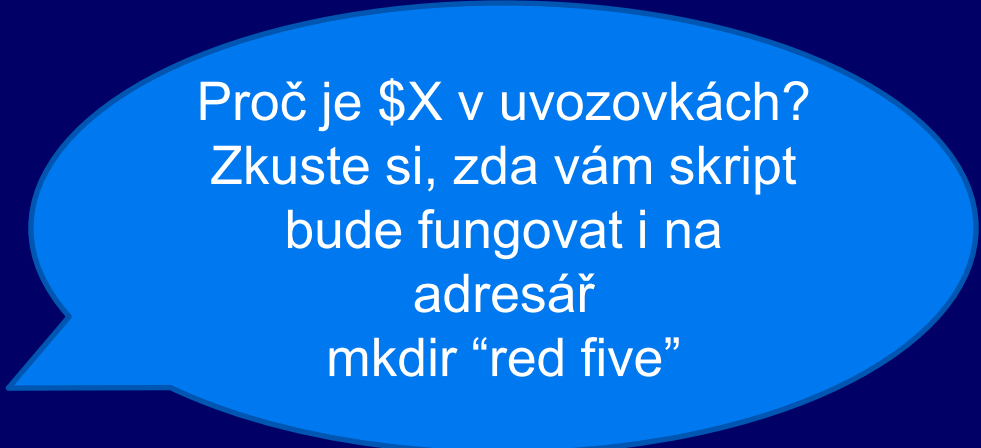


Př. for3

Jména podadresářů v aktuálním adresáři

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

```
for X in *  
do  
    if test -d "$X"  
    then  
        echo "Podadresar: $X"  
    fi  
done
```



Proč je \$X v uvozovkách?
Zkuste si, zda vám skript
bude fungovat i na
adresář
mkdir "red five"



Př. for4.sh

```
for soubor in *.txt
do
    echo vypis1: $soubor
done

for soubor2 in `ls`
do
    echo vypis2: $soubor2
done
```

iterace přes
soubory .txt
v aktuálním
adresáři

zpětné
apostrofy:
použije se
výstup
příkazu



Další možnosti for (for5.sh)

```
#!/bin/bash
for i in {1..5}
do
    echo "Ahoj cislo $i "
done
```

```
for i in {0..10..2}    .. toto jen některé verze bashe
seq 0 2 10           .. pokud není, toto lze použít
```



For s třemi výrazy (for6.sh)

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

```
for (( c=1; c<=5; c++ ))
```

```
do
```

```
    echo "Welcome $c times"
```

```
done
```



For – nekonečná smyčka (for7.sh)

```
#!/bin/bash
for (( ;; ))
do
    date
    echo "stiskni CTRL+C"
    sleep 2
done
```



Cyklus while, until

while seznam-prikazu

do seznam-prikazu

done

- dokud splněna podmínka (kód 0), cykluje
 - analogický příkaz **until** false ..
 - **break** – ukončení vnitřní smyčky
 - **continue** – vykonání smyčky
-



Operátory && a ||

binární operátory:

□ prikaz1 && prikaz2

- vyvolá prikaz1, pokud je návratová hodnota 0 (OK), vyvolá i prikaz2

□ prikaz1 || prikaz2

- vyvolá prikaz1, pokud je návrat. hodnota nenulová (neúspěch), vyvolá i prikaz2



Příklady

- `test -d rybnik || mkdir rybnik`
 - neexistuje-li adresář *rybnik*, vytvoříme jej
 - `test -d rybnik && mv ryba rybnik/ryba`
 - je-li adresář *rybnik*, přesuneme do něj soubor

 - vyzkoušejte z příkazové řádky:
 - `touch ryba ; mkdir rybnik ; atd...`
-

Vyhodnocení podmínky – []

```
if [ podmínka ]  
then  
    seznam-prikazu  
fi
```

Místo abychom psali

`test -d file`

Můžeme použít

`[-d file]`

Kolem hranatých
závorek

musí být

mezery !!!!

[] představuje příkaz, nutné mezery !!

if mezera [mezera podmínka mezera]



Podmínky

- ❑ -r soubor soubor přístupný pro **čtení**
- ❑ -w soubor soubor přístupný pro **zápis**
- ❑ -x soubor uživatel může soubor **spustit**
- ❑ -f soubor **obyčejný** soubor
- ❑ -d soubor **adresář**
- ❑ -c soubor **znakový** speciální
- ❑ -b soubor **blokový** speciální
- ❑ -p soubor pojmenovanou rourou
- ❑ -u soubor nastavený **set-user-ID bit**
- ❑ -g soubor nastavený **set-group-ID bit**
- ❑ -k soubor nastavený **sticky bit**



Podmínky2

- -z retezec pravdivé, když délka řetězce nulová
 - -n retezec pravdivé, když délka řetězce nenulová
 - $r1 = r2$ pravdivé, když identické
 - $r1 \neq r2$ pravdivé, když různé
 - retezec pravdivé, když neprázdný
-
- $n1$ relační op $n2$
 - -eq, -ne, -lt, -le, -gt, -ge
 - ! vyraz negace
 - vyraz1 -a vyraz2 oba pravdivé AND
 - vyraz1 -o vyraz2 alespon jeden OR
-



Počítací skript

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

```
N=1
```

```
while test "$N" -le "10"
```

```
do
```

```
    echo " Cislo $N "
```

```
    N=$((N+1))
```

```
done
```

Alternativa: $N=$((N + 1))$



Použití \$() a ``

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

```
ja=$(whoami)
```

```
ja2=`whoami`
```

```
echo "Ja jsem $ja"
```

```
echo "a taky $ja2"
```

```
echo dnes je $(date)
```

```
echo dnes je date
```

Spustí příkaz whoami, jeho výstup bude v proměnné ja, ja2

Spustí program date

Vypíše řetězec date



Sčítání čísel

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

```
a=5
```

```
b=2
```

```
c=$(( $a + $b ))
```

```
d=$(( $a / $b ))
```

```
echo soucet je: $c
```

```
echo celociselny podil je: $d
```

Příkaz `expr` lze nahradit
závorkami `()`

```
c=$((expr $x + $y))
```



Vnitřní proměnné shellu (!)

\$0	jméno skriptu
\$1, \$2, ...	poziční parametry skriptu,
\$*	seznam parametrů skriptu kromě jména skriptu,
\$#	počet parametrů,
\$\$	identifikační číslo procesu (PID) aktuálního SHELLu,
\$!	PID procesu spuštěného na pozadí,
\$?	návratový kód naposledy prováděného příkazu,
\$@	seznam parametrů ve tvaru "\$1" "\$2" "\$3" "\$4" .



Skript s parametry

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

```
echo "Nazev skriptu: $0"
```

```
echo "Pocet argumentu: $#"
```

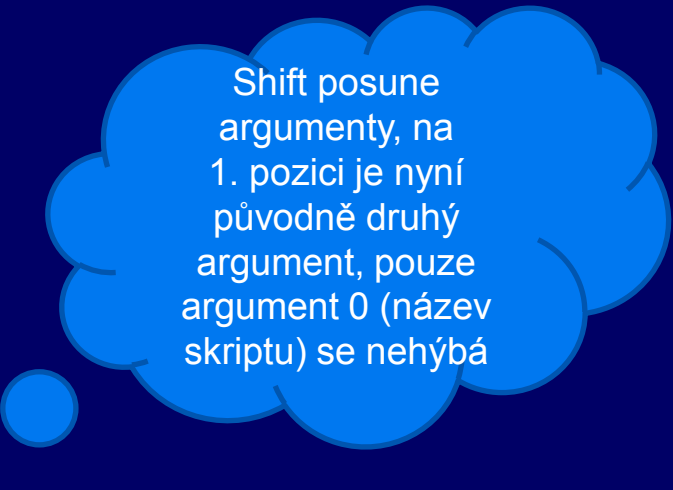
```
echo "Vsechny argumenty: $@"
```

```
echo "Prvni argument: $1"
```

```
echo "Paty argument: ${5}"
```

```
shift
```

```
echo "Druhy argument: $1"
```



Shift posune argumenty, na 1. pozici je nyní původně druhý argument, pouze argument 0 (název skriptu) se nehýbá



Skript s parametrem

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

```
# skript vyžaduje právě 1 argument
```

```
if [ "$#" -ne 1 ] ; then
```

```
    echo "Pouziti: $0 argument"
```

```
    exit 11
```

```
fi
```

```
echo Zadal jsi argument: $1
```

Středník
umožňuje then
na stejné řádce

Ukončení
skriptu s
návratovým
kódem 11



Skript opíše všechny parametry - while

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

```
while [ "$1" != "" ] ; do  
    echo "$1"  
    shift  
done
```



Skript opíše všechny parametry - for

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

```
for I in "$@"  
do  
    echo "$I"  
done
```

Iterace přes všechny parametry skriptu od \$1

Zkuste:
skript 1 2 3 "zeleny vlk"



Samostatná práce

Vytvořte následující skript

Název skriptu: setbit

Použití: `./setbit adresář`

Funkce: Skript vypíše z adresáře daného parametrem `adresář` všechny soubory, které mají nastavený set EUID bit

Např. soubor `/usr/bin/passwd` má nastavený set EUID bit



Co si mám přečíst?

- <http://www.root.cz/knihy/bash-ocima-bohdana-milara/>
 - Ucelená příručka o bashi v pdf
 - <http://www.abclinuxu.cz/clanky/ruzne/abcserialy#bash>
 - Další česky psaný materiál
 - <http://rute.2038bug.com/node10.html.gz>
 - A něco anglicky
-