

ZOS CVIČENÍ 4.

2024
L. Pešička



VYZKOUŠEJTE (OPAKOVÁNÍ)

- mc
- Stisknout CTRL+Z
- ps a
- fg
- top
- Stisknout CTRL+C

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
- `test -f adr1 && echo obycejny soubor`
- `test -f s1.txt && echo obycejny soubor`
- vyzkoušejte z příkazové řádky:
 - `touch ryba ; mkdir rybnik ; atd...`

VYHODNOCENÍ PODMÍNKY – []

```
if [ podmínka ]  
then  
    seznam-příkazu  
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 SCRIPT – POCET1.SH

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

```
N=1
```

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

```
do
```

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

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

```
done
```

Alternativa: `N=$((N + 1))` `.. N=$((vyraz))`

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 – POCITANI2.SH

```
#!/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))
```

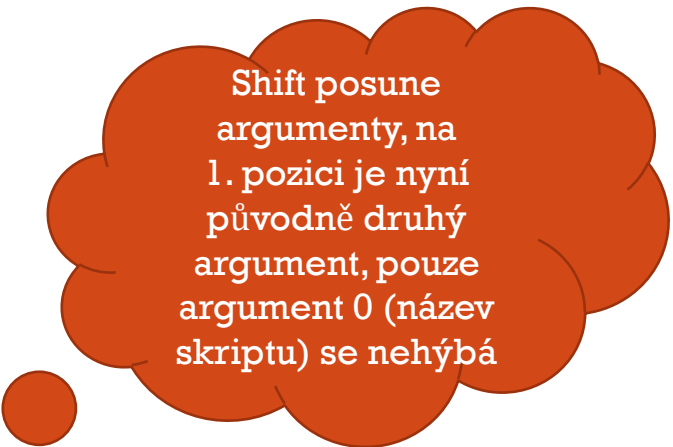
VNITŘNÍ PROMĚNNÉ SHELLU (!)

\$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 – PARA1.SH

```
#!/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 – PARA2.SH

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

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

```
if [ "$#" -ne 1 ] ; then  
    echo "Použiti: $0 argument"
```

```
    exit 11
```

```
fi
```

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

Středník
umožňuje then
na stejné řádce

Ukončení
skriptu s
návrátovým
kódem 11

SKRIPT OPÍŠE VŠECHNY PARAMETRY – PARA3.SH

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

```
while [ "$1" != "" ] ; do
```

```
    echo "$1"
```

```
    shift
```

```
done
```

SKRIPT OPÍŠE VŠECHNY PARAMETRY – PARA4.SH

```
#!/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

ZADÁNÍ: SKRIPT51.SH

Napište **skript51.sh** , který:

1. Vytvoří vždy soubor **autor.txt** obsahující login autora skriptu převedený na velká písmena
2. Při spuštění **s -a f** vytvoří symbolický link **slink** na soubor **f**
3. Při spuštění **s -b f** vypíše právě třetí řádku z **f** na obrazovku
4. Při spuštění **s -c f** nastaví práva na soubor **f**, že vlastník může vše, skupina jen číst, a ostatní nic
5. Při spuštění s jinými parametry vypíše text *Neplatný parametr!*

ZADÁNÍ: SKRIPT52.SH

- Skript má dva parametry
 - 1. parametr: adresář
 - 2. parametr: řetězec
- Skript bude procházet zadaný adresář (1.par) a vypíše všechny názvy souborů, které uvnitř souboru obsahují zadaný řetězec (2.par)

`./skript52.sh . poklad`

ŘEŠENÍ: SKRIPT52.SH

```
#!/bin/bash
if test "$#" -ne 2
then
    echo "Zadej $0 adresar retezec"
    exit 1
fi
for A in "$1"/*
do
    if test -f "$A" ; then
        if test -r "$A" ; then
            cat "$A" | grep "$2" > /dev/null
            if test $? -eq 0 ; then echo "$A" ; fi
        fi
    fi
done
```

využijeme testu návratové
hodnoty, zda předchozí příkaz
skončil úspěšně

WGET — STAHOVÁNÍ SOUBORŮ Z WEBU

- wget <http://www.zcu.cz>
 - Stáhne a uloží soubor index.html
- wget <http://nekde.cz/obrazek.jpg>
 - Stáhne soubor obrazek.jpg

VYUŽITÍ CYKLU FOR PRO STAHOVÁNÍ OBRÁZKŮ

```
for a in $( seq 9 )  
do  
wget http://www.neco.cz/obr${a}.jpg  
done
```

Stáhne obr1.jpg až obr9.jpg

Příkaz **seq x** generuje čísla 1 .. x

ZNAKOVÉ WWW PROHLÍŽEČE, MAIL

- links <http://m.idnes.cz>
- lynx <http://m.idnes.cz>
- pine – práce s poštou např. na eryxu
- mail pesicka@kiv.zcu.cz -s pozdrav
 - Ahoj, jak se mas? <Ctrl>+<D>
- mail pesicka@kiv.zcu.cz < povidka.txt

SPUŠTĚNÍ PROCESU NA POZADÍ

- `./vypocet &`
 - Spustí program výpočet na pozadí
 - V shellu můžeme zadávat další příkazy, nečeká na dokončení
- `nohup ./vypocet > vysledek.txt &`
 - Výpočet probíhá i po odpojení terminálu (např. zavřeme okno programu putty na eryx.zcu.cz)

NEZKOUŠET!

Pozor na nebezpečný příkaz !

Otestovat nanejvýš pouze v podadresáři!

- echo “ na tento prikaz pozor rm -rf * ”
 - Jen výpis
- echo “ na tento prikaz pozor ` rm -rf * ` “
 - **Nezkoušet, smaže!!**

Klasická ukázka **code injection**, kdy zdánlivě bezpečný příkaz echo může vést ke škodlivé činnosti

FUNKCE V BASHI — FCE1.SH

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

```
# prevzato z http://www.linuxexpres.cz/praxe/bash-23-dil
```

```
tento_pocitac()
```

```
{  
    echo -n "Pocitac: $HOSTNAME, cas: "  
    date  
}
```

```
echo "Obsazenost disku:"
```

```
tento_pocitac
```

```
df -h
```

```
echo
```

```
echo "Prihlaseni uzivatele:"
```

```
tento_pocitac
```

```
who
```

FUNKCE V BASHI – NÁVRATOVÁ HODNOTA – FCE2.SH

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

```
# prevzato z http://www.abclinuxu.cz/clanky/navody/bash-iv
```

```
vrat_retezec() {  
    echo "Řetězec"  
}
```

```
promena=$(vrat_retezec)
```

```
echo $promena
```

```
exit 0
```

FUNKCE — PARAMETRY — FCE3.SH

- Při volání funkce poziční parametry např. \$1 nahrazeny parametry funkce

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

```
obed() {
```

```
    echo "Mam chut na $1"
```

```
}
```

```
obed "pecene kure"
```

```
obed rizek
```

ITERACE PŘES VSTUPNÍ PARAMETRY

ITE1.SH

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

```
if [ $# -gt 3 ] ; then
```

```
    echo "Vic nez 3 parametry"
```

```
else
```

```
    echo "Nanejvys 3 parametry"
```

```
fi
```

```
for f in $@
```

```
do
```

```
    echo $f
```

```
done
```

AWK — MANIPULACE S TEXTEM

- `who | awk '{ print $1 }'`
- `who | awk '{ print $5, $1 }'`

Máme text. soubor `kont.txt`
(jmeno prijmeni tel)

Chceme vypsát ve tvaru (prijmeni jmeno tel)

- `awk '{ print $2, $1, $3}' kont.txt`
- `awk '{ print NR, $2, $1, $3}' kont.txt`



Číslo řádky

AWK

- `awk '/^Jan/ { print NR, $2,$1,$3}' kont.txt`
- `awk '/Novak/ { print NR, $2,$1,$3}' kont.txt`

- `/^Jan/` .. Vezme řádky začínající na Jan
- `/Zluta$/` .. Řádka končí na Zluta

- <http://cs.wikipedia.org/wiki/AWK>
- <http://www.ucw.cz/~hubicka/skolicky/skolicka20.html>

PROUDOVÝ EDITOR SED

- **sed /^#/d s1.txt**
 - Odstraní řádky, které začínají #
 - Mezi /... / je regulární výraz
 - ^ značí nový řádek
 - instrukce d – smaže řádek
- **sed '1,2d' s1.txt**
 - Smaže řádky 1-2

Vědět o něm,
že existuje

Více viz
cvičení
KIV/FJP

Literatura:

<http://melkor.dnp.fmph.uniba.sk/~zenis/prirucky/sed.html>

UKÁZKA SEDU

nahradi slovo obed slovem blaf

```
cat jidla.txt | sed -e s/obed/blaf/
```

na konec kazdeho radku prida tecku

```
cat jidla.txt | sed s/$/./
```

100 brambor zmeni na brambor 100

```
sed -r -e "s/([0-9]+)[[:space:]]+([[:alpha:]]+)/\2 \1/g"
```

POROVNÁNÍ OBSAHU SOUBORŮ

- **cmp** ceny.txt ceny2.txt
- **comm** ceny.txt ceny2.txt
- **diff** ceny.txt ceny2.txt

Pro aplikaci záplat se používá **patch**

Patch umí na základě souborů vytvořených diffem aplikovat změny do daného souboru

POUŽITÍ PATCHE

soubor **ceny.txt**

jablka 100
hrusky 50

soubor **ceny2.txt**

jablka 80
hrusky 50

```
diff ceny.txt ceny2.txt > mujpatch  
patch ceny.txt mujpatch
```

co bude nyní obsahovat soubor ceny.txt?
(apt-get install patch)

DOPLNĚNÍ A ROZŠÍŘENÍ PŘÍKAZŮ

ls -F	Označí ve výpise soubory dle typů např: adresar/ text.txt a.out*
ls -R	Vypíše rekurzivně obsah podadresářů
du -h	Využití diskového prostoru (kolik každý soubor zabírá)
df -h	Využití diskového prostoru filesystemů (kapacitu jednotlivých filesystemů, kolik je volno)
groups	Vypíše skupiny, kterých je uživatel členem
uptime	Jak dlouho systém běží

VZDÁLENÉ KOPIROVÁNÍ - SCP

```
scp ceny.txt pesicka@eryx.zcu.cz:~/
```

lokální soubor
ceny.txt

vzdálený stroj:
eryx.zcu.cz
uživatel na vzdáleném
stroji:
pesicka

domovský adresář
uživatele, tedy ~

zkopíruje soubor na vzdálený stroj

VZDÁLENÉ PROVEDENÍ PŘÍKAZU

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

```
for i in 201 202 203
```

```
do
```

```
    echo "prave vypinam pc s ip: 10.1.2.$i "
```

```
    ssh 10.1.2.$i /sbin/halt
```

```
done
```

KOPIROVÁNÍ A PROVEDENÍ VZDÁLENĚ

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

```
for i in 201 202 203 204
```

```
do
```

```
echo "prave instaluji iptables na pc s ip: 10.1.2.$i"
```

```
scp /root/ipt.acm 10.1.2.$i:/root/ipt.acm
```

```
ssh 10.1.2.$i iptables -P OUTPUT ACCEPT
```

```
ssh 10.1.2.$i iptables -F
```

```
ssh 10.1.2.$i iptables-restore < /root/ipt.acm
```

```
done
```

příkaz scp slouží ke kopírování souborů/adresářů mezi
lokálním a vzdáleným strojem

SUBSHELL

- vykonání operací v subshellu bez ovlivnění prostředí aktuálního shellu – (příkazy)
- `eryx2> pwd`
- `/users/p/pesicka/home`
- `eryx2> (cd /tmp ; pwd)`
- `/tmp`
- `eryx2> pwd`
- `/users/p/pesicka/home`

ZÁVORKY

- `{ pwd; ls; cd .. ; pwd; ls; } > vystup`
 - aktuální shell
 - Redirekce se týká výstupu ze všech příkazů
- `(pwd; ls; cd .. ; pwd; ls) > vystup`
 - Subshell
 - Zůstane ve svém adresáři

POZNÁMKA

- další možnosti použití závorek v bashi:

<http://stackoverflow.com/questions/2188199/how-to-use-double-or-single-bracket-parentheses-curly-braces>

POROVNÁNÍ ŘETĚZCŮ RETEZ1.SH

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

```
#test přihlaseného uživatele
```

```
echo "Zadej svoje přihlasovací jméno"
```

```
read JMENO
```

```
if test "$JMENO" = "$USER"
```

```
then
```

```
    echo "Zadal jsi tvoje přihlasovací jméno"
```

```
else
```

```
    echo "Zadaj jsi jiné jméno, než pod kterým jsi přihlasen"
```

```
fi
```

INTERAKTIVNÍ SKRIPT — SELECT.SH

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

```
select i in pivo vino ; do
    echo Odpovedel jsi: $i
    if [ -n "$i" ] ; then
        break
    fi
done
echo $REPLY
```

Zobrazí nabídku:

- 1) pivo
- 2) vino

Čeká na vstup uživatele,
dokud nezvolí jednu
z možností

\$i .. pivo nebo vino
\$REPLY .. 1 nebo 2

PŘESMĚROVÁNÍ SOUBORU CELÉMU CYKLU – CYKLUS1.SH

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

```
while read VSTUP
```

```
do
```

```
    echo "$VSTUP"
```

```
done < /etc/passwd
```

Bude číst postupně jednotlivé řádky ze souboru

POJMENOVANÁ ROURA

- `cd /tmp/pesi`
- **`mkfifo roura`** {nebo **`mknod roura p`**}
- `ls -l ; file roura`
 - **p**rw----- 1 pesicka users 0 2010-11-10 05:46 roura
- 1. Terminál jedna
 - `cat /etc/passwd > roura`
- 2. Terminál dva
 - `cat /tmp/pesi/roura`
 - `rm roura`

Prompt 1. terminálu se znovu neobjeví, dokud nepřečteme data z roury,
Stejně tak pokud napřed pustíme druhý terminál, blokuje, dokud nejsou data v rouře

ALIAS

- alias pozdrav='echo nazdarek'
 - Vytvoření aliasu
- pozdrav
 - Použití aliasu
- alias
 - Výpis aliasu
- unalias pozdrav
 - Zrušení aliasu
- alias ll='ls -l'