

echo

slová oddelené medzerou na štandardný výstup

```
echo 'Hello World'
```

cat

výpis obsahu súboru na štandardný výstup

```
cat /etc/passwd
```

WC

počet riadkov / slov / znakov / bajtov

```
wc -l /etc/passwd
```

head

prvých N riadkov súboru

```
head -n 10 /etc/passwd
```

tail

posledných N riadkov súboru

```
tail -n 10 /etc/passwd
```

od N -tého riadka do konca

```
tail -n +10 /etc/passwd
```

cut

vysekávanie políček podľa jednoznakového oddeľovača

```
cut -d: -f1 /etc/passwd
```

grep

vyhľadávanie a filtrovanie riadkov podľa regulár. výrazu

```
grep '^john' /etc/passwd
```

☐ -v riadky bez zhody

☐ -E rozšírený regex

☐ -i ignoruje VEĽKÉ/malé

awk

vylepšený cut s podporou viacerých oddeľovačov a pokročilých funkcií

```
awk -F: '{ print $1 }' /etc/passwd
```

- F znaky oddeľovačov
- \$1 prvé políčko v riadku

nl

očíslovanie riadkov

nl hello.c

sed

nahrádzanie textu v riadkoch

```
sed 's/root/admin/g' users.txt
```

- g** nahrádzanie všetkých výskytov v riadku
- E** zapne rozšírené regexy (GNU)

tr

nahrádzanie jednotlivých znakov, mazanie znakov

```
tr ' _ ' '- ' < files.txt
```

-d odstráni uvedené znaky

sort

triedenie podľa položiek

```
sort -t: -k3n /etc/passwd
```

-t oddeľovač políček

-k triedené políčko n číselné triedenie

uniq

zjednotí duplicitné riadky v zotriedenom vstupe

```
sort names.txt | uniq
```

printf

- vylepšené echo: podpora speci znakov
- formátovaný výpis textu

```
printf 'Pouzivatel %s byva v %s \n' john /home/john
```

Nový skript

- uvedený riadkom shebang

```
#!/bin/sh
```

- s atribútom executable

```
chmod +x skript.sh
```

POSIX Shell

- syntax shellu má milión dialektov
- POSIX: špecifikácia so zjednotenými vlastnosťami
 posixový skript pobeží všade
 (Linux, MacOS, AIX)
- bash: najrozšírenejší shell
- ksh, zsh, fish: ďalšie shelly

Podmienky

```
if exit kód príkazu je nula
```

```
then
```

```
...
```

```
else
```

```
...
```

```
fi
```

```
if grep root /etc/passwd
```

```
then
```

```
...
```

```
fi
```

Podmienky

```
if [ podmienka príkazu test ]  
then  
...  
else  
...  
fi
```

- f: je to súbor?
- d: je to adresár?
- n: neprázdna premenná
- z: prázdna premenná
- =: porovnanie

```
if [ -f /etc/passwd ] then
```


Premenné: čítanie

```
echo "$HOME"
```

- uvedená dolármi
- obalená úvodzovkami

Premenné: zápis

```
MENO='Grace Hopper'
```

- reťazce do apostrofov
- žiadne medzery okolo 

Premenné: z výstupu príkazu

```
USERS="$ (wc -l < /etc/passwd)"
```

- `$(...)` zachytí štandardný výstup príkazu
- uvedieme do úvodzoviek

Premenné a úvodzovky

- 'v apostrofoch' bežný reťazec
- "v úvodzovkách" reťazec, ale
\$, ', \ majú vlastný význam
- "\$HOME" čítanie z premenných
- "Domov: \$HOME" interpolácia
- 'wc -l' ekvivalent \$(wc -l)

Dostupné premenné

- `1`, `2`, atď: vstupné parametre
- `HOME`: domovský priečinok
- `LOGNAME`: login používateľa
- `PWD`: aktuálny adresár
- `PATH`: adresáre, kde sa hľadajú spustiteľné programy

Cyklus for

```
for X in slová oddelené bielym miestom  
do  
    echo "$X"  
done
```

Ak sa in slová oddelené bielym miestom vynechajú,
iteruje sa cez argumenty

Spracovanie súborov

```
for SUBOR in ./*.tex
do
  if [ -e "$SUBOR" ]
  then
    spracuj súbor v premennej SUBOR
  fi
done
```

- `./*` pre prípad súborov začínajúcich pomlčkou
`-e` lebo žolíky bez zhody expandujú sami na seba

Expanzia cesty

.

aktuálny adresár

..

rodičovský adresár

domovský priečinok

?

žolík pre jeden znak

*

žolík pre viacero znakov

[abcd]

množina znakov

find – vyhľadávanie v podadresároch

```
find . -name '*.c'
```

.

odkiaľ začať

-name '*.c'

podmienka

-name

hľadanie podľa mena

'*.c'

názov je v apostrofoch!

Je to argument pre find,
nie expanzia cesty!

Spracovanie súborov

wc s viacerými argumentami

```
find . -exec wc -l {} +
```

wc 1x pre každý súbor (staré, pomalé)

```
find . -exec wc -l {} \;
```

Riadky zo vstupu: xargs

```
seq 5 | xargs -I % touch 'file%.txt'
```

- Pre každý riadok zo vstupu sa vykoná príkaz.
- Znak `%` sa postupne nahrádza riadkom zo vstupu a vykonáva sa príkaz
- Častý zástupný znak: `{}` (à la find)

xargs folklór

Alternatíva pre **find/exec**:

```
find . | xargs -I % basename %
```

Spracovanie slov zo vstupu:

```
< mena.txt xargs printf '%s@bigcompany.com'
```

Cyklus while

```
while [príkaz s nulovým exit kódem]  
do  
...  
done
```

```
while sleep 3  
do  
    echo 'Ping!'  
done
```

Tipy pre hromadné spracovanie

for

súbory z jedného adresára,
postupnosť príkazov nad nimi

for

slová / parametre,
postupnosť príkazov nad nimi

find/exec

súbory zo stromu,
jeden príkaz

xargs

riadok/slovo zo stdin,
jeden príkaz nad ním

Načítanie riadkov: read

```
read -r LINE
```

- načíta do premennej LINE jeden riadok zo stdin
- ak sa riadok nenačíta, vráti nenulový exit kód
- parameter `-r` je vždy povinný

while/read

- načítavanie súboru po riadkoch
- `while` iteruje, kým `read` nevráti nenulový exit kód
- načítavame aj do viacerých premenných pre dáta oddelené medzerou
- konvencia: dáta nesmú ísť z rúry, ale súboru!

```
while read -r MENO PRIEZVISKO
do
    echo "$MENO, $PRIEZVISKO"
done < mena.txt
```

Funkcie

```
to_upper() {  
    echo "$1" | tr [:lower:] [:upper:]  
}
```

- `$1`, `$2`... argumenty funkcie
- argumenty sú stringové
- návratová hodnota: výhradne číselný exit kód (cez `return`)
- môže komunikovať cez `stdin/stdout/stderr`

Volanie funkcií

- funkcia je skript v skripte
- voláme ju bez zátvoriek

to_upper 'hello'

- presmerovanie výstupu do premennej takisto ako pri bežnom príkaze

MESSAGE="\$(to_upper 'hello')

Expanzie

`~` vlnky: domovský priečinok

`~` alebo `~root`

`$()` príkazu: zachytenie výstupu príkazu

`LOGIN=$(logname)`

`$(())` aritmetická: základná matematika

`I=$((I 1))`

`${ }` premennej: čítanie

`echo "${PATH}"`

Expanzie prázdných premenných

Ak je premenná 1 prázdna:

<code>\${1:-default}</code>	nahradí sa default hodnotou
<code>\${1:=default}</code>	priradí sa do nej default hodnota
<code>\${1:? 'Chyba premenna'}</code>	skript skončí s chybou a hláškou
<code>\${#1}</code>	dĺžka reťazca v premennej

Práca s reťazcami

predpis je slovo, ktoré môže obsahovať žolíky

- | | |
|----------------------------|---|
| <code>\${1%predpis}</code> | Odsekne najkratšiu príponu z konca |
| <code>\${1#predpis}</code> | Odsekne najkratšiu predponu zo začiatku |
| <code>\${1%%výraz}</code> | Odsekne najdlhšiu príponu |
| <code>\${1##výraz}</code> | Odsekne najdlhšiu predponu |

Składanie príkazov

- exit kódy možno považovať za true/false a skladať cez `&&` a `||`
- využíva sa skrátené vyhodnocovanie
- `&&`: príkaz spusti, len ak predošlý príkaz uspel
- `||`: ak príkaz zlyhá, spusti nasledovný príkaz

Skladanie príkazov

- oznám zlyhanie

```
grep "^alice" /etc/passwd || echo "Ziadna Alice"
```

- založ adresár, ak neexistuje

```
[ -d ./cache ] || mkdir ./cache
```

- zmaž súbor, ak existuje

```
[ -f .lock ] && rm .lock
```

Triky s &&

príkaz1 && príkaz2

príkaz2 sa vykoná, len ak príkaz1 uspeje

Príkaz 1	&&	Príkaz 2	=	Výsledok
OK	&&	OK	=	OK
FAIL	&&	nevykoná sa	=	FAIL
OK	&&	FAIL	=	FAIL

Triky s ||

príkaz1 || príkaz2

príkaz2 sa vykoná, ak príkaz1 zlyhá

Príkaz 1		Príkaz 2	=	Výsledok
OK		nevykoná sa	=	OK
FAIL		OK	=	OK
FAIL		FAIL	=	FAIL

Zoznamy príkazov

```
príkaz1;príkaz2
```

2 príkazy na jednom riadku

```
príkaz1\  
    príkaz2
```

2 príkazy v jednom

```
{príkaz1; príkaz2; }
```

Viacero príkazov sa tvári ako jeden pri presmerovaní vstupov výstupov

Subshell

- shell spustí samostatný shell
- zdedia sa deskriptory súborov
- skopírujú sa premenné
 - zmeny premenných sa neprejavia v rodičovskom shelli
 - zmeny premennej v rúre sa neprejavia u rodiča

Subshelly nastanú:

(príkaz1; príkaz2)	2 skripty v izolácii
príkaz1 príkaz2	spustenie príkazov v rúre
X=\$(príkaz)	zachytenie príkazu do premennej

Awk

Pre každý riadok spĺňajúci **predpis** sa vykoná **akcia**

```
predpis { akcia }
```

Spustenie:

```
awk -F ':' '{print }'
```

```
awk -F ':' -f skript.awk
```

-F: oddeľovač políček

Predpisy awk

/regex/	{..}	riadok spĺňa regex
NR=3	{..}	tretí riadok
\$3 > 3	{..}	tretia položka > 3
\$1 ~ /OK/{..}		prvá položka spĺňa regex
BEGIN	{..}	pred prvým riadkom
END	{..}	po poslednom riadku
NR>3,/OK/{..}		kombinácia

Akcie awk

<code>{ print }</code>	vytlačí celý záznam/riadok
<code>{ print \$1 }</code>	vytlačí prvú položku
<code>{ print \$3, \$1 }</code>	3. a 1. položka oddelené výstupným oddeľovačom (medzera)
<code>{ print "*" \$3 }</code>	konkatenácia medzerou

Zabudované premenné awk

\$0	celý riadok
\$1, \$2 atď	obsah položiek na aktuálnom riadku
NR	poradové číslo riadka
IFS	oddeľovač políček (viď '-F')
OFS	oddeľovač políček na výstupe
NF	počet položiek v riadku

Premenné awk

IFS=","	printf IFS
MESSAGE="Hello"	print HELLO
COUNT=0	print COUNT

awk rozpoznáva reťazce v úvodzovkách, čísla
a asociatívne polia

Funkcie awk

`gsub(čo, čím, kde)`

Nahradenie reťazca v celom riadku

`sprint("format", parametre...)`

formátovanie a priradenie

`split(reťazec, do_poľa)`

rozsekne reťazec do cieľového poľa

`getline`

načíta ďalší riadok

Programowanie awk

Cyklus:

```
for (i = 0; i < NF; i++) { print i }
```

Podmienka:

```
if ( COUNT > 0 ) { print "OK" }
```

Programovanie awk

Funkcia

```
function sucet(x,y) return x + y  sucet(2+3)
```

Polia:

```
a["John"] = 1  
a[0] = 1
```

sed – spúšťanie

sed **program** súbor

program priamo v riadku

sed -e **program** **-e** **program** *atd'* súbor

viacero programov

sed -f **program v súbore** súbor

externý skript s programom

sed -n

zruš implicitný výpis riadkov

[s]ubstitute - nahrad'

s/č**o**/č**ím**/g

č**o** BRE regex. Pozor na obmedzenú syntax!

č**ím** BRE regex

g nahrádzanie všetkých výskytov

s/pes/dog/ - nahrad' prvý výskyt

s/;/;/g - hromadné nahradenie

[s]ubstitute - nahrad'

s/[0-9]//g
Odstráň čísla

[] čo je BRE regex, **[]** čím môžeme vynechať.

s/pes/+&+/g
Obal' pluskami

[] & reprezentuje nájdený text.

s#-###/g

Oddeľovač je mriežka.

s#.* \(.*\)#\1#g
*Nechaj len 2.
slovo*

Skupiny uzatvárame do escapovaných zátvoriek. Odkaz na 1. skupinu **[]** \1

Adresy

adresa1,adresa2 príkaz

adresa:

- číslo riadku. Posledný riadok: \$
- /regex/

Príkazy podľa typu berú 0, 1 alebo 2 adresy.

[p]rint - tlač

p	tlač každý riadok 2x (raz implicitne, raz explicitne)
sed -n 1p	len 1. riadok. Implicitný výpis vypnutý
sed -n 1,5p	prvých 5 riadkov (=head)
sed -n 3,\$	od 3. riadku do konca
sed -n /#/p	len riadky s # (=grep)

[d]elete - maž

1,3d

vymaže prvé 3 riadky

6,\$d

vymaže od 6. riadku do konca
nechá prvých 5 riadkov

/#/d

vymaže riadky začínajúce #

[i]nsert, [a]ppend, [c]hange

/public class/[i] /* @author jp */
vloží pred riadok daný text

/public class/[a] /* class */
vloží za riadok daný text

1,3[c] —
zamení prvé tri riadky za čiaru

/^#/ [c] —
zamení riadok začínajúci mriežkou za čiaru

Viacriadkové skripty pre sed

```
sed -e 's/pes/dog/' -e 's/vlk/wolf/'
```

Riadok postupne putuje príkazmi.

Externé skripty pre sed

```
1,3 {  
    s/pes/dog/  
    s/vlk/wolf  
}
```

V súboroch. Zavádzame parametrom -f

```
sed -f skript.sed
```

sed - zriedkavé príkazy

= čísloje riadky

n načíta ďalší riadok

N prilepí ďalší riadok k aktuálnemu