# BASH

- promjena ljuske → chsh -s bash
- pomoć → ls --help
  → man ls
- brisanje datoteke → rm ime
  kazala → rmdir ime
- append → ls >> ime_dat
- stdin → program < ime_dat
- cjevovod → prog1 | prog2
- ako želim 21.-35. redak → tail -n +20 <lista.txt | head -n 15 >kratka.txt
- uvjetno izvođenje → who | grep -s "mario" && "Mario je tu"
  who | grep -s "mario" || "Mario nije tu"
- varijable: inicijaliziraju se normalno, bez razmaka S="bok",
  dohvaćaju preko $S

PWD=$PWD"/ciklus1" <=> cd ciklus 1

$USER - kor. ime
$SHELL - trenutna ljuska (uključujući i put)

Koraci obrade retka:
1. ljuska čita
2. razlaganje na niz tokena
3. parsiranje - jednostavne i složene naredbe
4. širenje riječi - rezultat - lista imena putanja
   - poja naredba i argumenata
5. obavlja se preusmjeravanje (i uklanjaju operatori)
6. pokreće se fja
7. ljuska čeka izlazni status

- znakovi za escapeanje: ! & ; < > ( ) $ ` \ " '
- '$PWD' → string "ovo je $PWD" → varijabla, jer se šire argumenti
- `naredba` → umjesto nje dolazi njen izlaz
- širenja: ① vitičaste zagrade
  ② ~
  ③ parametri i varijable
  ④ zamjena naredbi
  ⑤ aritmetičko
  ⑥ dijeljenje riječi
  ⑦ širenje imena datoteka

① širenje vitica : a{b,c} postaje ab ac
in use : mkdir projekt/{old, new, bugs}

② ~ : ako ~riječ → moguće korisničko ime
ako ~ → $HOME kazalo onog ko je pokrenuo ljusku

③ varijable i parametri → ${ab} - vitice da se izbjegne moguća pogreška
→ ${varijabla:-word} - korištenje default
→ ${varijabla:=word} - postavljanje default

④ naredbe : $(naredba) ⟺ `naredba`
↳ u ovom obliku unutarnji navodnici s "\"

⑤ aritmetički izrazi - zamjena aritmetičkih izraza rezultatom
- članovi mogu imati druga širenja
- oblik : $((aritm_izraz))

⑥ imena datoteka - nakon dijeljenja na riječi, traže se *, ?, [
↳ ako se pronađe → uzorak
↳ u njemu se posebni znakovi escapeaju
↳ uzorak se zamjenjuje abecednom listom imena datoteka
↳ * → podniz(i ε), ? → 1 znak, [...] → treba s jednim znakom

☺ može se zadati raspon: [a-c]
☺ ako je prvi znak unutar [] ^ ili ! onda negacija
☺ ako treba "-", onda prvi ili zadnji
☺ ako treba "[", mora biti zadnji
☺ klase [[:klasa:]] → alpha, blank, digit, space, upper, lower

---

WC wc [opcije] [<datoteka>] → ako nema datoteke → stdin (pipeline)

grep grep [opcije] trazeni_uzorak [<datoteka>] → pretraživanje datoteka
↳ "<cv.h>" ili korisnik ili regex → ugl je 'regex'

regexi → lista znakova []: ako 1. znak ^ → negacija!
: [a-c] može biti [abc] ili [aBbCc] ovisno o env!

→ klase [:alnum:] ili \w, [^[:alnum:]] ili \W, [:alpha:], [:digit:], ...
→ ] mora na početku
→ ^ ne smije na počeku
→ - mora na kraju
→ "." 1 znak
→ ^ početak retka
→ $ kraj retka npr. grep '.\eps$' *.txt

# Regexi (cont'd)

→ "\<" početak "\>" kraj riječi

→ "\b" granica riječi

→ "\B" sve osim granice riječi

Operatori ponavljanja: ? 0..1 {n} n
* 0..∞ {n,} n..∞
+ 1..∞ {n,m} n..m

→ ulančavanje podniza

→ može i ILI fja preko "|"

→ default: ponavljanje > ulančavanje > alternacija

↓ ako nema zagrada

→ backreference: \broj na grupu unutar uokvirenih zagrada

**sort** sort [opcije] [<datoteka>]

↳ redak po redak

**uniq** izbacuju se samo uzastopna ponavljanja

**sed** sed = stream editor

- zamjena teksta sed 's/stari/NOVI' dat.txt
- zamjena svih pojavljivanja sed 's/stari/NOVI/g' dat.txt
- brisanje uzorka sed 's/... $//' dat.txt
- ispis redaka u kojima se pronađe uzorak sed -n '/UNIX/p' dat.txt
- brisanje redaka s uzorkom sed '/UNIX/d' dat.txt

**find**

```
find /etc -size +1M    (>1MB)
find /etc -size -1k    (<1kB)
find ~/ -mtime 0 -ls   (ispiši sve mijenjane danas)
find ~/ -mtime -3      (u zadnja 3 dana)
find . -name '*.c' -print   (NE obavlja se širenje!)
```

**locate** locate [<opcije>] <uzorak>, uzorak pod '', -r regex, -i case insensitive

---

ako se prenosi preko okoline → source skripta.sh

nije preko okoline → ./skripta.sh → možemo export a napravit

**parametri**
- $0 → ime ljuske/ime skripte → podljuska
- $n ili ${11} npr

| | |
|---|---|
| shift 1 | $1 → $0 |
| broj par | $# |
| svi par | $@ |

$? →izlazni status zadnje naredbe

**uvjet**

```
if naredba_t
then
    blok_naredbi
fi
```

```
if naredba_1 ; then
    blok_1
elif nar_2 ; then
    blok_2
else
    blok_3
fi
```

**test** ide kao naredba if

može test naredba <=> [naredba ]

npr. [ "$name" = "Vedrana" ]

ovo treba, jel ako je neinicijalizirano nestane

uoči!!

test -n "a" ✓
test -z "" ✓

usporedbe : =eq, != ne, <lt, ≤le, >gt, ≥ge

: nizovi niz1 = niz2

[! ] →negacija izraza

-a AND

-o OR

"\(" ... "\)" za mijenje redoslijeda

**exit**

→exit n
→default: exit $?

**case**

```
case value in
    pat_1) blok_1;;
    pat_2) blok_2;;
    pet_n) blok_n;;
esac
```

**for**

```
for var in rijec1 rijec2 ... rijecn
do
    blok
done
```

→radi se širenje imena dat: for i in *.sh

→for arg in "$@" <=> for arg

→čitanje iz datoteke: for file in $( cat lista.txt); do echo $file; done

**while**

```
while test_uvjet
do
    blok
done
```

**until** → dok se to ne ispuni!

```
until test_uvjet
do blok
done
```

**break**
→ break
→ break n ⇐ n unutrašnjih petlji

**continue**
→ continue
→ continue n : preskače naredbe u n petlji

**redirekcija u petlji**

```
for bla in bla
do blok
done <lista.txt > nova.txt
```

**read**

```
read prva_rijec druga_rijec ostatak_retka
while read Redak ; do ...
```

**printf**

```
printf "Oktalna vrijednost broja %d je %o\n" 20 20
```

# PERL

skalar → broj (255, 3.25e20) } $count != $count
$a_1
→ string

→ broj - literal: 037, 0x37, 0b37, 0x50-37
grupiranje

→ operatori: 10 \ 3 #= 3.333 ... FP!
2 ** 4 #= 2 ^ 4

→ string: a) u 'niz' : escapea se samo \' i \\ (\n ostaje npr.)
b) "niz" : • interpolacija varijabli
• \n, \t, ...

c) operacije: "hello" . "world" # "helloworld"
"hello" . ' ' . "world # 'hello world'
'hello world' . "\n" # "hello world\n"
"bu" x (3+1) # "bububu"
5 x 4 #"5555"

d) "12abc34" * "3" # 36 ; "z" . 5*7 # "z35"

operacije: može $a += 1;

**print**

```
print "Rezultat je ", 3*7, ".\n";
print "Reza je ${bok}c.\n";
```

↳pazi je sam $bokc.\n će uzet $bokc

usporedbe: može i == i eq

**if**

```
if (uvjet) {
    blok
}
```

⇒vitice moraju

(! ) → negacija!

undef, 0, '', '0' false, svi ostali true!

**<STDIN>**

```
$line = <STDIN>;
```

⇒$line sadrži \n!

```
$broj = chomp <STDIN>
```

↳ 0 ili 1, koliko ih je

-prije inicijalizacije, varijabla je undef, a to je ⇒ ""

**while**

funkcija defined($redak) undef su EOF ili Ctrl+D

**liste** lista je skup vrijednosti

polje je var u kojoj je lista (ne mora postojati)

prvi element liste → indeks 0

**polja**

```
$fred[0] = "bla";
$fred[1] = "blaić";
```

} to je polje, a možemo istovremeno imati i skalar $fred=5

automatski se polje proširuje, mogu biti i nastati undef elementi

preko $#polje se može kontrolirat veličina pojа, a varijabla je zadnji indeks!

$polje[-n] → od kraja, ne smije previše

**liste**

(1.7..5.7) = (1,2,3,4,5) NEMA (5..2)

(0,2..6,8,10)

($a . $b)

(0 .. $#polje)

qw/ bla bla2 / → kao da su u ' '

\escape

jednostavni swap

višak se zanemaruje ili undef