

0. Kratke upute za rad u ljusci na UNIX-u/Linuxu

Natuknice

- računalo za rad: bilo koji UNIX/Linux sustav, pravi i virtualni, laptop, ...
 - preporuka: ako već nemate neki sustav s UNIXom/Linuxom, koristiti VMware/VirtualBox ili slično, ili
 - ako imate windows 10/11 možete koristiti i Windows Subsystem for Linux
 - * upute kako to ostvariti na webu
 - * kako doći do datoteka koje su tamo?
 - * pokrenuti explorer iz komandne linije: `explorer.exe . &`
- spajanje i rad u “terminalu”: lokalni terminal, putty, ssh, ...
- osnovne naredbe: `ls`, `pwd`, `cd`, `mkdir`, `cp`, `mv`, `rm` (!)
- ljuska: `bash` (tab, strelice)
- tekstualni editor: `pico` (`nano`, `vi`, `vim`)
- prevođenje (kompajliranje): `gcc` (`g++`, `cc`, `CC`)
- pokretanje programa: `./a.out`
- rad s procesima: `ps`, `kill -9 PID`
- (info) rad s resursima: `ipcs`, `ipcrm`
- (info) prijenos datoteka: `copy/paste`, `scp`, `winscp`, `dos2unix`, `unix2dos`

Primjer 0.1. Primjer programa, prevođenje, pokretanje

Napisati program, pohraniti ga pod imenom `vj1.c`, prevesti (kompajlirati) i pokrenuti ga.

```
$ pico vj1.c
```

Upisati program (paziti na stil pisanja: razmaci, uvlačenja, zagrade, prozračnost!)

```
1 #include <stdio.h>
2 int main()
3 {
4     printf("Hello World!\n");
5     return 0;
6 }
```

Izlaz iz editora: `Ctrl + x`, `yes`, `yes`

```
$ gcc vj1.c                // program se pokreće sa "./a.out"
$ gcc vj1.c -o prog        // program se pokreće sa "./prog"
```

Ostale zastavice: `-O[0-3]` `-Wall` `-Werror`

Česte greške na labosima

- `ime.c != ime.C`
 - prevoditelj (gcc) prepoznaje prvo (`.c`) kao C, a drugo (`.C`) kao C++ program
 - `.cpp .cxx .cc .C .CC  $\Rightarrow$  C++`
- Segmentation fault, Core dumped, Bus error
 - kazaljka pokazuje izvan procesnog adresnog prostora
 - najčešći razlozi: neinicijalizirana kazaljka; neinicijalizirana varijabla koja se koristi kao indeks polja
- funkcije pozivati sa zagradama ()
 - pri prepisivanju pseudokoda u kod, ako se poziva funkcija (i bez argumenata) obavezno staviti zagrade
 - npr. u pseudokodu piše: `zabrani_prekidanje;`
a u implementaciji postoji funkcija: `void zabrani_prekodanje() { ... }`
onda ju u kodu pozvati sa `zabrani_prekodanje();`
 - ako se izostave zagrade, prevoditelj neće javiti grešku, ali se funkcija neće pozvati!

Stil pisanja koda

- kod najčešće nije namijenjen samo osobi koja ga stvara
- u praksi će taj kod dijeliti s drugima, koji će surađivati/nastaviti s radom
- također, često se ne kreće od nule već od postojećeg koda
- na FER-u taj kod još gledaju asistenti
- stoga kod MORA biti pisan prema uobičajenim pravilima za taj programski jezik
- postoji nekoliko sitnih varijacija pravila za C, odabrati bilo koju i konzistentno ju se držati
- preporuka je koristiti pravila koja se koriste za [kod jezgre Linuxa](#)
- osnovni zahtjev: čitljivost – struktura koda mora biti vidljiva
 - uvlačenje koda: “tab” ili razmaci (preporuka: tabulator)
 - KONZISTENTNO koristiti tab ILI razmake
 - NE miješati tab i razmake
 - podesiti alat koji se koristi (editor) na jedno ili drugo i konstantno ga koristiti (ili sve koji se koriste podesiti na isto)
- poželjno:
 - komentirati kod (preporuka, ne uvjet)
 - lomljenje predugih redova radi čitljivosti (npr. 80./100. stupac)
 - “prozračnost” koda – umetanje praznih linija i razmaka
 - razmaci: `a__b__+_c__*_d;` ne `a=b+c*d;`
 - zagrade `()`, `[]`, `{}`: npr. `if__(a[4][2]_==_5) {`
(zagrada `{` u novi red samo za funkcije ili iznimno zbog preglednosti)