VEŽBA 4

Pregled

Viši programski jezici, naročito jezik C, podržava podelu izvornog koda složenijeg softverskog projekta na module, tj. jedinice koje se prevode odvojeno, a rezultat se na kraju procesa spaja u konačan oblik. Ovo značajno usložnjava proces prevođenja, pa je obavljanje procesa prevođenja direktnim zadavanjem komandi naporno i podložno greškama. Zato se to obično automatizuje, a jedno rešenje za to je alat `make` koji je predmet ove vežbe.

Make alat – `makefile`

Make alat svoj posao obavlja na osnovu podataka u tzv. makefile-u čije je podrazumevano ime upravo `makefile`. Centralni deo ovog fajla čini opis relacija zavisnosti oblika:

```
target: <dependency list>
 -TAB- <recipe commands>
```

`rarget` je fajl čija se zavisnost opisuje. Zavisi od svih fajlova navedenih u listi `dependency list` koju čine imena fajlova odvojeni praznim mestom.

Deo `recipe commands` predstavlja shell komande koje će se izvršiti ukoliko je barem jedan element iz `dependency list`-a noviji od `target`-a. Takav slučaj znači da postoji potreba za ažuriranjem fajla u kom treba da se odraze izmene napravljene u fajlovima od kojih zavisi. Sve linije (može ih biti više) `recipe commands` obavezno počinju karakterom TAB.

Pozivanje kompajlera za jezik C – `gcc`

Kada je u pitanju jezik C, prevođenje se obavlja u dve faze.

Prevođenje programskog modula u objektni modul

Svaki programski modul koji je napisan u jeziku C i upisan u fajl sa tekstom programa (*izvorni kod*, *engl. source code*) analizira se i prevodi se bez posebnog osvrtanja na ostale programske module i zapisuje u *objektni modul*. Ipak, buduća interakcija sa drugim modulima će biti neophodna, a informacije o tim vezama sadržana je u, često brojnim i često hijerarhijski povezanim, heder fajlovima koje programski modul uključuje korišćenjem makro instrukcije `#include`.

Komanda kojom se poziva alat za prevođenje (kompajler) u cilju kreiranja objektnog modula je sledeća:

```
gcc -c -O1 -Wall <ime_fajla_sa_izvornim_kodom>
```

Iako nije obavezno, skoro bez izuzetka fajl sa izvornim kodom ima ekstenziju `.c` za programe pisane u jeziku C. Opcija `-O1` uključuje optimizacije nivoa 1, dok opcija `-Wall` zahteva ispisivanje svih upozorenja kompajlera. Jednim pozivanjem kompajlera, moguće je prevesti samo jedan programski modul. Ime fajla koji će sadržati objektni modul biće isto kao i ime fajla sa izvornim kodom, jedino što će ekstenzija biti promenjena u `.o`.

Povezivanje objektnih modula u izvršni modul

Svi objektni moduli u sebi sadrže mašinski kod programa i veze prema ostalim modulima sa kojima će činiti konačan program. Konačan program je po pravilu sastavljen iz više modula i njih je neophodno povezati u konačan izvršni modul koji je spreman za primenu na ciljnom sistemu.[1]

Komanda kojom se postiže povezivanje objektnih modula je sledeća:

```
gcc -o <ime_izvrsnog_modula> <lista_objektnih_modula>
```

Iako je komanda ista kao i za prevođenje pojedinačnih modula, alat koji obavlja ovaj posao je nešto drugačiji i naziva se **linker**-om (*engl. linker*). Nakon opcije `-o` treba zadati ime izvršnog modula. Komanda se završava listom imena objektnih modula odvojenih zarezom koji će se povezati u izvršni modul. Iako nije obavezno, skoro uvek objektni moduli imaju ekstenziju `.o`.

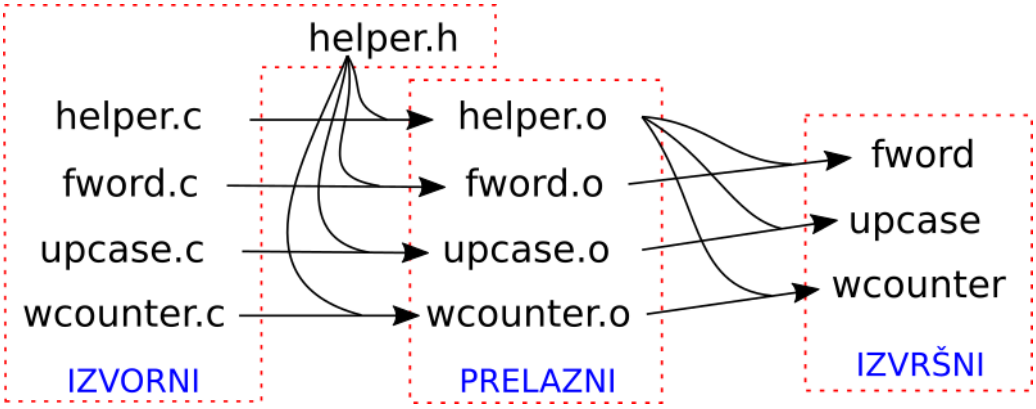
Zadaci

Za izradu zadataka, neophodno je preuzeti arhivu fajlova vezba_04.tar.gz i raspakovati je u radnom direktorijumu. To se može uraditi jednom komandom

```
tar xf vezba_04.tar.gz
```

Arhiva sadrži početni `makefile`, izvorne kodove test programa kao i tekstualne fajlove za brzo testiranje prevedenih programa.

Međusobna zavisnost fajlova za ovu vežbu predstavljena je sledećom šemom:



1. Iskoristiti dati `makefile` i pomoću njega uraditi prevođenje programa `upcase`. Ovaj program konvertuje tekst sa standardnog ulaza u tekst koji sadrži samo velika slova. Za primenu na tekstualne fajlove, najsvrsishodnije je koristiti preusmeravanje poput

   ```
   upcase < z128.txt
   ```

2. Dopuniti `makefile` u skladu sa datom šemom tako da `make` bude u stanju da proizvede program `wcount`. Testirati program pomoću nekog tekstualnog fajla. Program procenjuje broj reči u tekstu, obratiti pažnju na način vraćanja rezultata.
3. Dopuniti `makefile` u skladu sa datom šemom tako da `make` bude u stanju da proizvede program `fword`. Testirati program pomoću nekog tekstualnog fajla. Program utvrđuje da li se

1 Osim sadržaja programskih modula, u sastav izvršnog modula obično ulaze i neki sistemski moduli sadržani u tzv. bibliotekama.

zadata reč nalazi unutar teksta koji je primljen preko standardnog ulaza. Npr.

```
fword automobil < z1300.txt
```

Obratiti pažnju na način na koji program vraća rezultat. Uraditi testiranje sa više ulaznih fajlova i više različitih reči koje se traže.

4. Napisati shell skript koji korisniku omogućava izbor preko menija (`select`) sledećeg oblika

```
1)brojanje
2)pretraga
```

Prva opcija omogućava nalaženje ukupnog broja reči u svim tekstualnim fajlovima vidljivim u tekućem direktorijumu (`*.txt`), dok druga omogućava unos proizvoljne reči koja će se tražiti u istim tekstualnim fajlovima kao i kod prethodne opcije.

Podsetnik: U oba slučaja bi za obradu fajlova sa zadatom šemom imena trebalo razmisliti o korišćenju `for` petlje pri čemu se lista elemenata za obradu generiše korišćenjem odgovarajućih džoker karaktera.

Za one koji bi da eksperimentišu: ako hoćemo da pretraga ne bude osetljiva na velika i mala slova (prihvata reči sa proizvoljnom kombinacijom velikih i malih slova) potrebno je i traženu reč i tekstualni fajl prevesti u velika slova i tek onda uraditi pretragu. Dopuniti drugu opciju tako da ispunjava i ovaj zahtev.