Programiranje 2
Programski jezik C
Datoteke,
Argumenti komandne linije
2. čas

Mirko Spasić

DATOTEKE

- U C-u se sa fajlovima barata na sledeci nacin:

  u standardnoj biblioteci postoji definisana struktura:

```
typedef struct {
....
} FILE;
```

koja sadrzi sve informacije vezane za otvoreni fajl. Ove informacije su platformski zavisne, i nama nisu od znacaja.

- Njima barataju funkcije standarnde biblioteke.

DATOTEKE

- Fajl se otvara funkcijom:

  FILE * fopen(char *ime, char *nacin);

  Ova funkcija prihvata string sa putanjom do fajla (ime), kao i string (nacin) koji definise način rada sa fajlom. Može biti:

  "r" -- fajl se otvara za čitanje (mora da postoji)

  "w" -- fajl se otvara za pisanje (kreira se ako ne postoji, a ako postoji, njegov sadržaj se briše)

  "a" -- fajl se otvara za dopisivanje (kreira se ako ne postoji. Prilikom svakog pisanja sadržaj se dodaje na kraj fajla).

DATOTEKE

- Funkcija fopen() vraca pokazivač na FILE strukturu koja je povezana sa otvorenim fajlom. Od tog trenutka se baratanje sa otvorenim fajlom sastoji u tome da se funkcijama standardne biblioteke prosledi ovaj pokazivač kao argument. Ostalo obavlja biblioteka za nas.
- Ako fajl nije uspešno otvoren, povratna vrednost f-je je NULL. Ova konstanta je definisana u stdio.h zaglavlju na sledeći način:

```
#define  NULL  0
```

- Dakle, u pitanju je nula. Uobičajeno je da se ova konstanta koristi za pokazivače koji imaju vrednost nula (tj. ne pokazuju ni na šta).

DATOTEKE

- Prilikom svakog otvaranja fajla, treba proveriti da li je nastupila greška, tj. da li je povratna vrednost funkcije bila NULL. Ako jeste, treba nekako obraditi tu grešku. Najčešći način za to je obavestiti korisnika o tome šta se desilo i prekinuti program.

- Nakon završetka rada sa fajlom, treba ga zatvoriti:

```
fclose(f);
```

gde je f FILE pokazivač.

- Prilikom završetka rada programa, svi fajlovi se automatski zatvaraju. Mi se nećemo oslanjati na ovo, već kad god nam otvoreni fajl više ne treba, eksplicitno ćemo ga zatvarati.

DATOTEKE

- Standardni ulaz i izlaz se u C-u takođe svodi na ulaz i izlaz fajova. Prilikom pokretanja programa se automatski otvaraju tri fajla, kojima se dodeljuju sledeći globalni FILE pokazivači:

  - stdin -- standardni ulaz (povezan sa tastaturom)
  - stdout -- standardni izlaz (povezan sa ekranom)
  - stderr -- standardni izlaz za greske (povezan sa ekranom)

DATOTEKE

- Još neke funkcije standardne bibilioteke koje barataju sa fajlovima:

```
int fgetc(FILE *f);
```

Učitava jedan karakter iz fajla na koji pokazuje f, i vraća njegov ASCII kod kao povratnu vrednost. Ako je nastupio kraj fajla, vraća EOF.

```
int fputc(int c, FILE *f);
```

Upisuje karakter c u fajl na koji pokazuje f. Vraća c.

- Imajući ovo u vidu, lako je uočiti da su sledeće naredbe ekvivalentne:

```
getchar() <=> fgetc(stdin)
putchar(c) <=> fputc(c, stdout);
```

DATOTEKE

char *fgets(char *s, int size, FILE *f);

Učitava celu liniju karaktera iz fajla na koji pokazuje f u string s, ali ne više od size karaktera (što treba da bude veličina stringa s). Eventualni '\n' sa kraja linije se čuva u stringu s, koji se ovom f-jom I terminira nulom. F-ja vraća NULL ako se desila greška pri učitavanju.

int fputs(const char *s, FILE *f);

Upisuje string s u fajl na koji pokazuje f bez terminirajuće nule. Vraća EOF za grešku.

DATOTEKE

int fscanf(FILE *f, const char *format, ...);

int fprintf(FILE *f, const char *format, ...);

Ove dve funkcije rade isto što i scanf i printf, osim toga što ne rade sa standardnim ulazom, tj. sa standardnim izlazom već rade sa fajlom na koga pokazuje f.

long ftell(FILE *f);

Ova f-ja vraća tekuću poziciju datoteke f ili -1L u slučaju greške.

Ako nam zatrebaju detaljniji opis neke od ovih funkcija možemo ga naći u man stranicama za tu funkciju. U konzoli kucamo
man fprintf

ako tražimo dodatni opis funkcije fprintf.

DATOTEKE

int fseek(FILE *f, long offset, int origin);

Funkcija fseek podešava poziciju datoteke f. Narednim čitanjem ili pisanjem pristupa se podacima na početku nove pozicije. Nova pozicija se postavlja na offset znakova od mesta origin, koje može biti SEEK_SET (početak datoteke), SEEK_CUR (tekuća pozicija) ili SEEK_END (kraj datoteke). Za tekstualnu datoteku, offset mora biti nula ili vrednost koju je vratila funkcija ftell i u tom slučaju origin mora biti SEEK_SET.

Funkcija fseek vraća vrednost različitu od nule u slučaju greške.

Primer: fseek(f, 0, SEEK_END) postavlja datoteku na 0 bajtova od kraja datoteke, tj. na kraj datoteke.

ARGUMENTI KOMANDNE LINIJE

Prilikom pozivanja programa iz komandne linije moguće je navesti određene opcije i argumente, koje program koristi u svom radu. Npr, možemo pozvati program na sledeći način:

```
./izvrsni   ulaz.txt   izlaz.txt
```

zeleci da program izvrsni iskopira fajl ulaz.txt u fajl izlaz.txt.

Potrebno je da postoji mehanizam da program iznutra pristupi stringovima koje je korisnik naveo prilikom poziva programa.

U C-u se to ostvaruje na sledeći način:

ARGUMENTI KOMANDNE LINIJE

Definišemo parametre funkcije main:

- int argc - ovaj parametar prilikom pokretanja programa dobija vrednost broja argumenata koje je korisnik prilikom poziva naveo. Pri tom se i ime programa broji kao argument komandne linije. U gornjem primeru će argc biti postavljeno na 3.

- char **argv - pokazivač na pokazivač na char koji sadrži adresu prvog u nizu pokazivača na karaktere. Svaki od tih pokazivača na karaktere sadrži adresu odgovarajućeg stringa koji sadrži argument komandne linije. Npr. u gornjem primeru, argv[0] će nam pokazivati na string "./primer", argv[1] će nam pokazivati na string "ulaz.txt", argv[2] će nam pokazivati na string "izlaz.txt".

Dakle, niz pokazivača na čiji prvi član pokazuje argv je dužine upravo argc.

ARGUMENTI KOMANDNE LINIJE

- Uobičajena praksa na UNIX sistemima je da se argumenti komandne linije dele na opcije i argumente u užem smislu (kraće argumenti).
- Opcije počinju znakom '-' nakon čega obično sledi jedan karakter koji označava koja je opcija u pitanju. Ovim se najčešće upravlja funkcionisanjem programa i neke mogućnosti se uključuju ili isključuju. Argumenti najčešće predstavljaju opisne informacije poput imena fajla, ili podatke nekog drugog tipa.

```
gcc -o izvrsni primer.c
```

U prethodnom pozivanju prevodioca, -o je opcija, a izvrsni i primer.c su argumenti u užem smislu.

ZADACI

1. Napisati program koji prepisuje datoteku ulaz.txt u datoteku izlaz.txt i to (a) karakter po karakter (b) liniju po liniju.

2. Napisati program koji ispisuje broj navedenih argumenata komandne linije, a zatim i same argumenate sa rednim brojevima.

3. Napisati program koji prepisuje datoteku čije se ime navodi kao prvi argument komandne linije u datoteku čiji se argument navodi kao drugi argument komandne linije programa. Ukoliko je navedena opcija -u program prilikom prepisivanja treba da zamenjuje sva mala slova velikim, a ukoliko je navedena opcija -l sva velika slova se zamenjuju malim. Ako nisu navedeni argumenti koji predstavljaju imena datoteka, raditi sa standardnim ulazim, tj. izlazom.