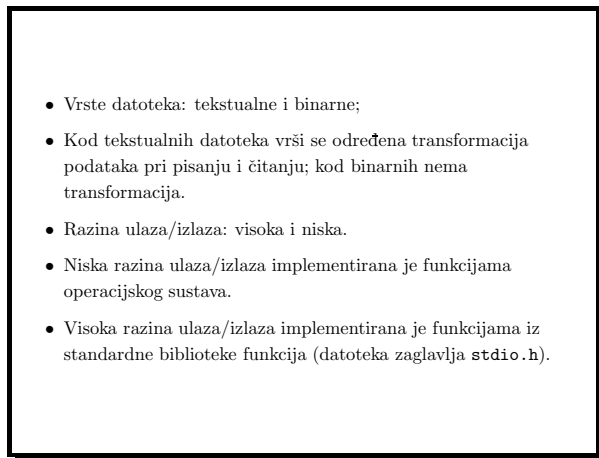
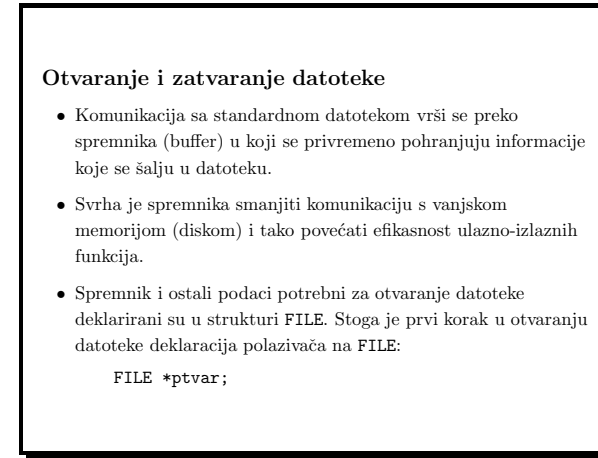


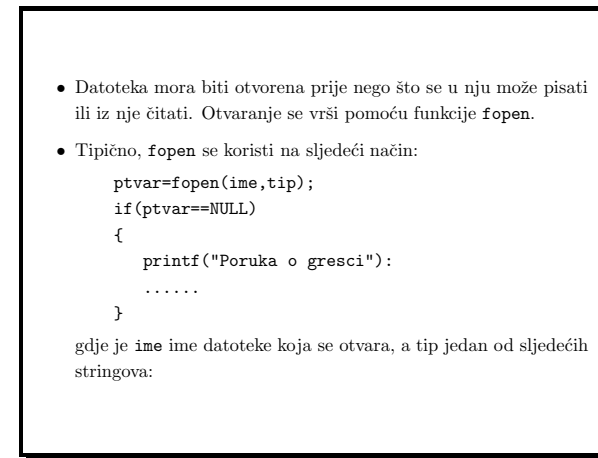
Slide 1



Slide 2



Slide 3



Slide 4

tip	Značenje
"r"	Otvaranje postojeće datoteke samo za čitanje
"w"	Kreiranje nove datoteke samo za pisanje.
"a"	Otvaranje postojeće datoteke za dodavanje teksta.
"r+"	Otvaranje postojeće datoteke za čitanje i pisanje.
"w+"	Kreiranje nove datoteke za čitanje i pisanje.
"a+"	Otvaranje postojeće datoteke za čitanje i dodavanje teksta.

- Ako se postojeća datoteka otvori s `w` ili `w+` njen sadržaj će biti izbrisan i pisanje će početi od početka.
- Ako datoteka koju otvaramo s tipom `a` ili `a+` ne postoji bit će kreirana, a ako postoji novi tekst će biti dodavan na kraju datoteke (eng. append).

Slide 5

Binarne datoteke: tipu se dodaje `b` koji označava binarno.

- "`rb`", "`wb`", "`ab`" = binarno čitanje, pisanje, dodavanje;
- "`rb+`" ili "`r+b`" = binarno čitanje;
- "`wb+`" ili "`w+b`" = binarno pisanje;
- "`ab+`" ili "`a+b`" = binarno dodavanje.

UNIX ima samo jedan tip datoteka (binarno=tekstualno).

Slide 6

- Funkcija `fopen` vraća pokazivač na strukturu `FILE` povezanu s datotekom.
- Funkcija vraća `NULL` ako datoteka nije mogla biti otvorena.
- Na kraju programa datoteka treba biti zatvorena funkcijom `fclose` koja uzima kao argument pokazivač na `FILE`:

```
fclose(ptvar);
```

Primjer:

```
#include <stdio.h>

FILE *fpt;
if((fpt=fopen("primjer.dat", "w"))==NULL)
    printf("\nGRESKA: Nije moguće otvoriti datoteku.\n");
.....
fclose(fpt);
```

Slide 7

- Svakom programu stoje na raspolaganju tri automatski otvorene "datoteke": standardni ulaz, standardni izlaz i standardni izlaz za greške.
- Standardni ulaz je tastatura računala, standardni izlaz i standardni izlaz za greške također ekran računala.
- Pod operacijskim sustavom UNIX standardni ulaz i izlaz mogu se preusmjeriti pri pozivu programa tako da učitavanje/ispis vrše iz datoteke tj. u datoteku.
- U datoteci `<stdio.h>` definirani su konstantni pokazivači na `FILE` strukturu povezanu sa standardnim ulazom, izlazom i izlazom za greške. Ti pokazivači imaju imena `stdin`, `stdout` i `stderr`.

Slide 8

Funkcije za čitanje i pisanje (1)

- Funkcije

```
int getc(FILE *fp);
```

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

vraćaju sljedeći znak iz datoteke na koju pokazuje fp.

- Razlika između `getc` i `fgetc` je u tome da `getc` može biti implementirana kao makro naredba dok `fgetc` ne smije.
- U slučaju greške ili kraja datoteke vraća se EOF.
- Funkcija `getchar()` implementira se kao `getc(stdin)`.

Primjer:

Slide 9

Funkcije za čitanje i pisanje (2)

- Funkcije

```
int putc(int c, FILE *fp);
```

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

upisuju znak c u datoteku na koju pokazuje fp i vraćaju upisani znak.

- Razlika između `putc` i `fputc` je u tome da `putc` može biti implementirana kao makro naredba dok `fputc` ne smije.
- U slučaju greške vraća se EOF.
- Funkcija `putchar(c)` implementira se kao `putc(c, stdout)`.

Primjer:

Slide 11

```
#include <stdio.h> /* Broji znakove u datoteci */
#include <stdlib.h> /* Ime datoteke je argument
                    komandne linije */
int main(int argc, char *argv[]) {
    int ch,i=0;
    FILE *fpt;
    if(argc==1){ /* ime datoteke nedostaje */
        printf("\nUporaba: a.out ime_datoteke\n");
        exit(-1);
    } else if((fpt=fopen(argv[1],"r"))==NULL){
        printf("Ne mogu otvoriti datoteku %s\n",argv[1]);
        exit(-1);
    }
    while((ch=fgetc(fpt))!=EOF) i++;
    fclose(fpt); printf("\nBroj znakova = %d\n",i);
}
```

Slide 10

Funkcija koja kopira jednu datoteku u drugu.

```
void cpy(FILE *fpulaz, FILE *fpizlaz)
{
    int c;

    while((c=getc(fpulaz))!=EOF)
        putc(c,fpizlaz);
}
```

Slide 12

Funkcije za čitanje i pisanje (3)

Funkcija koja učitava podatke iz datoteke liniju-po-liniju je

```
char *fgets(char *buf, int n, FILE *fp);
```

- `buf`= pokazivač na dio memorije (*buffer*) u koji će ulazna linija biti spremljena; `n`= veličina memorije na koju prvi argument pokazuje; `fp`= pokazivač na datoteku iz koje se učitava.
- Funkcija će pročitati liniju uključujući i znak za prijelaz u novi red i na kraj će dodati nul znak `'\0'`. Pri tome će biti učitano najviše `n-1` znakova, uključivši `'\n'`.
- Ukoliko je ulazna linija dulja od toga, ostatak će biti pročitán pri sljedećem pozivu funkcije `fgets`.
- Funkcija vraća `buf` ako je sve u redu ili `NULL` ako se došlo do kraja datoteke ili se pojavila greška.

Slide 13

fgets i gets:

- Funkcija

```
char *gets(char *buf);
```

čita sa standardnog ulaza.

- `gets` ne uzima veličinu *buffera* kao argument i stoga se može desiti *pretek buffera*.
- Umjesto `gets(buf)` treba koristiti `getf(buf,n,stdin)`.
- Pri tome treba uzeti u obzir da `fgets` učitava i `'\n'`, dok `gets` znak za prijelaz u novi red ne učitava već na njegovo mjesto stavlja `'\0'`.

Slide 14

Funkcije za čitanje i pisanje (4)

- Funkcija za ispis podataka u datoteku liniju-po-liniju je

```
int fputs(const char *str, FILE *fp);
```

- Funkcija vraća nenegativnu vrijednost ako je ispis uspio ili EOF u slučaju greške.
- `fputs` ispisuje znakovni niz na koji pokazuje `str` u datoteku na koju pokazuje `fp`. Zadnji nul znak neće biti ispisan.
- Funkcija

```
int puts(const char *str);
```

ispisuje znakovni niz na koji pokazuje `str` na standardni izlaz. Na kraj niza ona dodaje znak za prijelaz u novi red što ju razlikuje od `fputs(str,stdout)`.

Slide 15

Formatirani upis/ispis:

- Za formatirani ispis u datoteku i upis iz datoteke koristimo funkcije

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

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

identične sa `printf` i `scanf`, s time da im je prvi argument pokazivač na datoteku.

- `fscanf` vraća broj učitanih objekata ili EOF ako je došlo do greške ili kraja datoteke; `fprintf` vraća broj upisanih znakova ili negativan broj u slučaju greške.
- `printf(...)` je ekvivalentno s `fprintf(stdout,...)`, a `scanf(...)` je ekvivalentno s `fscanf(stdin,...)`.

Slide 16

Greške:

Budući da ulazne funkcije vraćaju EOF i u slučaju kada je došlo do greške i u slučaju kada se naiđe na kraj datoteke postoje funkcije

```
int ferror(FILE *fp);
int feof(FILE *fp);
```

koje razlikuju te dvije situacije.

- **ferror** vraća broj različit od nule (istina) ako je došlo do greške, a nulu (laž) ako nije.
- **feof** vraća broj različit od nule (istina) ako smo došli do kraja datoteke, a nulu (laž) u suprotnom.

Primjer:

Slide 17

Binarni upis/ispis:

Funkcije za binarno čitanje i pisanje.

```
size_t fread(void *ptr, size_t size,
             size_t nobj, FILE *fp);
```

```
size_t fwrite(const void *ptr, size_t size,
             size_t nobj, FILE *fp);
```

One ne vrše konverziju iz binarnog u ASCII format i obratno tj. direktno se upisuje (čita) binarni zapis u (iz) datoteku(e).

- **fp** = pokazivač na datoteku u koju se piše ili iz koje se čita.
- **ptr** = pokazivač na varijablu u koju **fread** upisuje odnosno iz koje **fwrite** čita.
- **nobj** = broj objekata koje treba ispisati/učitati.
- **size** = veličina pojedinog objekta.

Slide 19

```
#include <stdio.h> // kopiranje std. ulaza na
const int MAXLINE=128; // std. izlaz

int main(void) {
    char buf[MAXLINE];

    while(fgets(buf,MAXLINE,stdin) != NULL)
        if(fputs(buf,stdout)==EOF) {
            fprintf(stderr, "\nIzlazna greska.\n");
            exit(1);
        }
    if(ferror(stdin)) {
        fprintf(stderr, "\nUlazna greska.\n");
        exit(2);
    }
}
```

Slide 18

- Funkcije **fread** čita iz datoteke na koju pokazuje **fp** niz od **nobj** objekata dimenzije **size** i smješta ih u polje na koje pokazuje **ptr**.
- Vrijednost koju funkcija vraća je broj učitanih objekata koji je manji od **nobj** ako je došlo do greške ili kraja datoteke.
- Funkcije **fwrite** upisuje u datoteke na koju pokazuje **fp** niz od **nobj** objekata dimenzije **size** iz polja na koje pokazuje **ptr**.
- Vrijednost koju funkcija vraća je broj upisanih objekata koji je manji od **nobj** ako je došlo do greške.

Primjer:

Slide 20

```

FILE *fpt;
struct account{
    short count;
    long total;
    char name[SIZE];
} d11;

fpt=fopen(...);
.....
if(fwrite(&d11,sizeof(d11), 1, fpt) != 1){
    fprintf(stderr,"Greska pri upisu\n");
    exit(1);
}

```

- Prednost binarnog ulaza/izlaza je brzina i veličina zapisa.
- Nedostatak binarnog ulaz/izlaz je zavisnost o arhitekturi računala i prevodiocu.

Slide 21

Direktan pristup podacima

Pomoću do sada uvedenih funkcija podacima možemo pristupiti samo sekvencijalno. Sljedeće dvije funkcije nam omogućavaju direktan pristup.

```

int fseek(FILE *fp, long offset, int pos);
long ftell(FILE *fp);

```

- Funkcija `ftell` uzima kao argument pokazivač na otvorenu datoteku i vraća trenutni položaj unutar datoteke. U slučaju greške vraća `-1L`.
- Funkcija `fseek` uzima pokazivač na otvorenu datoteku, te dva parametra koji specificiraju položaj u datoteci. Treći parametar `pos` indicira od kog mjesta se mjeri `offset` (odmak). Funkcija tekuću poziciju u datoteci smješta na zadanu lokaciju.

Slide 22

U datoteci `stdio.h` su definirane tri simboličke konstante: `SEEK_SET`, `SEEK_CUR` i `SEEK_END`, koje se mogu koristiti kao treći argument funkcije `fseek`, a koje imaju sljedeće značenje:

pos	Nova pozicija u datoteci
<code>SEEK_SET</code>	offset znakova od početka datoteke
<code>SEEK_CUR</code>	offset znakova od trenutne pozicije u datoteci
<code>SEEK_END</code>	offset znakova od kraja datoteke

Na primjer,

Slide 23

Poziv	Značenje
<code>fseek(fp, 0L, SEEK_SET)</code>	Idi na početak datoteke
<code>fseek(fp, 0L, SEEK_END)</code>	Idi na kraj datoteke
<code>fseek(fp, 0L, SEEK_CUR)</code>	Ostani na tekućoj poziciji
<code>fseek(fp, ftell_pos, SEEK_SET)</code>	Idi na poziciju je dao prethodni poziv <code>ftell()</code> funkcije

- Standard postavlja neka ograničenja na funkciju `fseek`. Tako za binarne datoteke `SEEK_END` nije dobro definiran, dok za tekstualne datoteke jedino su pozivi iz prethodne tabele dobro definirani.

Slide 24

Primjer. Napišimo sada funkciju koja će u datoteci potražiti određeni `account` i povećati njegovo polje `total` za 100, ako je ono manje od 5000.

```
void Povecaj(const char *file, int n)
{
    FILE *fp;
    struct account tmp;
    long    pos;
    int     size=sizeof(struct account);

    if((fp=fopen(file,"r+b"))==NULL){
        fprintf(stderr,"Povecaj, error: nije moguće otvoriti "
            " datoteku %s.\n",file);
        exit(-1);
    }
}
```

Slide 25

```
    pos = n*size;
    fseek(fp, pos, SEEK_SET);
    if(fread(&tmp, size, 1, fp) != 1){
        fprintf(stderr,"Greska pri citanju.\n");
        exit(1);
    }
    if(tmp.total < 5000L) {
        tmp.total += 100;
        fseek(fp, -size, SEEK_CUR);
        if(fwrite(&tmp, size, 1, fp) != 1){
            fprintf(stderr,"Greska pri upisu.\n"); exit(1);
        }
    }
    fclose(fp);
    return;
}
```

Slide 26

Funkcija mijenja sadržaj `n+1`-og zapisa (brojanje zapisa počinje od nule). Prvo se otvara datoteka za čitanje i upisivanje (`r+b`), a zatim se pokazivač pozicionira na početak `n+1`-og zapisa (`fseek(fp, pos, SEEK_SET)`);

Zapis se pročita i ako je `total` manji od 5000 povećava se za 100. Nakon što je zapis promijenjen, treba ga upisati nazad u datoteke. U tu svrhu se prvo vraćamo na početak `n+1`-og zapisa (`fseek(fp, -size, SEEK_CUR)`);

i onda vršimo upis.

Slide 27