



Operativni sistemi 2013

Sistemska programiranje Datotečni sistem

**Katedra za računarstvo
Elektronski fakultet u Nišu**

**Prof. drDragan Stojanović
mr Aleksandar Stanimirović
mr Bratislav Predić**



Sadržaj



- Funkcije za rad sa datotekama
- Funkcije za rad sa direktorijumima



Sadržaj



- Funkcije za rad sa datotekama
- Funkcije za rad sa direktorijumima



Funkcije za rad sa datotekama

Otvaranje datoteke

```
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
int open (char * filename, int flags [, mode_t mode]);
```

Semantika

- Sistemski poziv **open** se koristi za otvaranje postojeće datoteke za čitanje ili pisanje odnosno za kreiranje nove datoteke.
- Prvi argument funkcije **filename** je znakovni niz koji sadrži puno ime datoteke zadato kao apsolutna ili relativna putanja.



Funkcije za rad sa datotekama

Semantika

- Drugi argument **flags** se dobija kao **razultat OR operacije** nad sledećim konstantama:
 - ▶ **O_RDONLY** – datoteka se otvara samo za čitanje.
 - ▶ **O_WRONLY** – datoteka se otvara samo za upis.
 - ▶ **O_RDWR** – datoteka se otvara za upis i za čitanje.
 - ▶ **O_APPEND** – pozicionira pokazivač datoteke na kraj. Datoteka je spremna za upis na kraj.
 - ▶ **O_CREAT** – ukoliko datoteka sa zadatim imenom ne postoji, kreira se nova.
 - ▶ **O_TRUNC** – ukoliko datoteka sa zadatim imenom postoji njen sadržaj se briše (datoteka postaje dužine nula)
- Treći argument **mode** je opcioni i koristi se prilikom kreiranja nove datoteke za definisanje privilegija. Ukoliko se izostavi koriste se podrazumevane privilegije koje zavise od toga koji je korisnik pokrenuo proces.
- U slučaju uspeha sistemski poziv vraća pozitivnu celobrojnu vrednost koja predstavlja **deskriptor datoteke** a u slučaju greške vraća -1.
- Svakoј otvorenoj datoteci u sistemu je pridružen deskriptor datoteke.



Funkcije za rad sa datotekama

Zatvaranje datoteke

```
#include <unistd.h>  
int * close (int fd);
```

Semantika

- Sistemski poziv **close** **zatvara datoteku** koja je prethodno otvorena korišćenjem sistemskog poziva **open**.
- Arguemnt **fd** predstavlja deskriptor datoteke koja se zatvara.
- Po izlasku iz programa sistem automatski zatvara sve otvorene datoteke.



Funkcije za rad sa datotekama

```
#include <stdio.h>
#include <stdlib.h>
int main(int argc, char * argv[])
{
    int fd;
    fd = open("test", O_RDONLY);
    fd = open("test", O_RDWR | O_CREAT, 00777);
    fd = open("test", O_WRONLY | O_TRUNC);
    if (fd < 0)
    {
        printf("Doslo je do greske");
        exit(1);
    }

    close(fd);
    exit(0);
}
```

Deskriptor datoteke.

Datoteka se otvara samo za čitanje.

Datoteka se otvara za čitanje i pisanje. Ukoliko ne postoji kreira se.

Datoteka se otvara samo za čitanje. Prethodni sadržaj se briše.

Datoteka se na kraju zatvara.



Funkcije za rad sa datotekama

Čitanje i pisanje iz datavoda

```
#include <unistd.h>
ssize_t read(int fd, void * buff, size_t count);
ssize_t write(int fd, void * buff, size_t count);
```

Semantika

- Sistemski pozivi koji se koriste za **čitanje** odnosno **pisanje podataka** iz datoteke.
- Prvi argument predstavlja **deskriptor odgovarajuće datoteke**.
- Drugi argument predstavlja pokazivač na bafer iz koga se podaci upisuju u datoteku ili u koji se podaci upisuju iz datoteke.
- Treći argument predstavlja **broj bajtova** koji se upisuju u datoteku ili se čitaju iz datoteke.
- Povratna vrednost funkcija predstavlja broj pročitanih bajtova odnosno broj upisanih bajtova u datoteku. U slučaju greške funkcija vraća -1.



Funkcije za rad sa datotekama

Pozicioniranje pokazivača datoteke

```
#include <unistd.h>
#include <sys/types.h>
off_t lseek(int fd, off_t offset, int mode);
```

Semantika

- Sistemski pozivi koji se koriste za **pozicioniranje pokazivača datoteke** na proizvoljnu poziciju u datoteci.
- Prvi argument **fd** predstavlja **deskriptor odgovarajuće datoteke**.
- Drugi argument **offset** definiše novu poziciju unutar datoteke a tumači se na osnovu vrednosti trećeg argumenta.
- Treći argument **mode** definiše kako se određuje nova pozicija:
 - ▶ **SEEK_SET** – offset je definisan u odnosu na početak datoteke.
 - ▶ **SEEK_CUR** – offset je definisan u odnosu na trenutnu poziciju pokazivača datoteke.
 - ▶ **SEEK_END** – offset je definisan u odnosu na kraj datoteke.
- Povratna vrednost funkcija predstavlja novu poziciju u datoteci U slučaju greške funkcija vraća -1.



Funkcije za rad sa datotekama

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <fcntl.h>

//Maksimalna velicina bafera
#define MAX_BUFF[256]

int main(int argc, char * argv[])
{
    int fd;
    char buff[MAX_BUFF];
    int i;
    int n;

    //Mora postojati bar jedna datoteka
    if (argc < 2)
    {
        printf("Nema dovoljno argumenata\n");
        exit(1);
    }
```

```
//Obraduju se sve prosledene datoteke
for (i = 1; i < argc; i++)
{
    fd = open(argv[i], O_RDONLY);

    //Došlo je do greške pa se prelazi na narednu datoteku
    if (fd < 0)
        continue;

    //Sadržaj datoteke se prikazuje na standardnom izlazu
    printf("Ime datoteke: %s\n", argv[i]);

    //Podaci se očitavaju u petlji
    while((n = read(fd, buff, MAX_BUFF)) > 0)
        printf("%s", buff);

    close(fd);
}
```



Funkcije za rad sa datotekama

Informacije o datoteci

```
#include <unistd.h>
#include <sys/types.h>
int stat(char * name, struct stat * buff);
```

Semantika

- Sistemski pozivi omogućava **pribavljanje informacija** o nekoj datoteci.
- Prvi argument **name** predstavlja **ime datoteke** koje je zadato kao apsolutna ili relativna putanja.
- Drugi argument **buffer** predstavlja pokazivač na strukturu koja će prihvatiti informacije o datoteci.



Funkcije za rad sa datotekama

Struktura stat

```
#include <sys/stat.h>
struct stat {
    dev_t      st_dev;      /* broj uređaja */
    ino_t      st_ino;      /* broj inode strukture */
    mode_t     st_mode;     /* dozvole pristupa datoteci */
    nlink_t    st_nlink;    /* broj linkova ka ovoj datoteci */
    uid_t      st_uid;      /* identifikacija korisnika */
    gid_t      st_gid;      /* identifikacija grupe */
    dev_t      st_rdev;     /* tip uređaja */
    off_t      st_size;     /* veličina datoteke u bajtovima */
    blksize_t  st_blksize;  /* veličina blokova za U/I datotečnog sistema */
    blkcnt_t   st_blocks;   /* broj dodeljenih blokova */
    time_t     st_atime;    /* vreme poslednjeg pristupa datoteci */
    time_t     st_mtime;    /* vreme poslednje modifikacije datoteke */
    time_t     st_ctime;    /* vreme poslednje promene statusa */
};
```



Funkcije za rad sa datotekama

Makroi za određivanje tipa datoteke

● Makro naredbe koje za argument uzimaju atribut **st_mode** iz strukture **stat** a vraćaju TRUE ili FALSE u zavisnosti od tipa datoteke:

- ▶ **S_ISDIR** – direktorijum
- ▶ **S_ISREG** – regularna datoteka
- ▶ **S_ISCHR** – znakovna specijalna datoteka
- ▶ **S_ISBLK** – blok specijalna datoteka
- ▶ **S_ISFIFO** – imenovani datavod (pipe) ili FIFO
- ▶ **S_ISLNK** – link
- ▶ **S_ISSOCK** – socket



Sadržaj



- Funkcije za rad sa datotekama
- Funkcije za rad sa direktorijumima



Funkcije za rad sa direktorijumima

Otvaranje direktorijuma

```
#include <dirent.h>  
DIR * opendir(const char * pathname);
```

Semantika

- Sistemski pozivi omogućava **otvara postojeći direktorijum**.
- Prvi argument **pathname** predstavlja **ime direktorijuma** koje je zadato kao apsolutna ili relativna putanja.
- U slučaju greške sistemski poziv vraća **NULL**.
- U slučaju uspeha sistemski poziv vraća pokazivač na **DIR strukturu**. Ova struktura se koristi u svim sistemskim pozivima za rad sa direktorijumima kada je potrebno **identifikovati otvoreni direktorijum**.



Funkcije za rad sa direktorijumima

Čitanje sadržaja direktorijuma

```
#include <dirent.h>
struct dirent * readdir(DIR * dp);
```

Semantika

- Sistemski pozivi omogućava **čitanje sadržaja otvorenog direktorijuma**.
- Prvi argument **dp** predstavlja pokazivač na otvoreni direktorijum.
- U slučaju uspeha sistemski poziv vraća pokazivač na **dirent strukturu** koja sadrži informacije o stavki direktorijuma.
- Pokazivač **dp** se **automatski inkrementira** da pokazuje na sledeću stavku u direktorijumu. Kako bi obradili sve stavke u nekom direktorijumu treba **sukcesivno pozivati** funkciju **readdir** dok funkcija ne vrati NULL vrednost. Funkcija vraća NULL vrednost kada pokazivač dp dodje do kraja direktorijuma.



Funkcije za rad sa direktorijumima

Stavka direktorijuma

● Sadržaj ove strukture zavisi od implementacije operativnog sistema i obavezno ima dva člana:

```
struct dirent {  
    ino_t  d_ino;  
    char  d_name[NAME_MAX+1];    /* ime stavke koja se nalazi u direktorijumu,  
                                   može biti poddirektorijum, datoteka ili veza  
                                   (link) */  
};
```



Funkcije za rad sa direktorijumom

Resetovanje direktorijuma

```
#include <dirent.h>  
void rewinddir(DIR * dp);
```

Zatvaranje direktorijuma

```
#include <dirent.h>  
void close(DIR * dp);
```



Funkcije za rad sa direktorijumom

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/file.h>
#include <sys/dirent.h>
#include <sys/stat.h>
#include <sys/types.h>
#include <sys/fcntl.h>
int main(int argc, char * argv[])
{
```

```
    DIR *dp;
    struct dirent *dirp;
```

Otvaranje direktorijuma.

```
    if ( (dp = opendir("test")) == NULL)
    {
        printf("Ne moze se otvoriti direktorijum");
        exit(1);
    }
```

Obrada stavki iz direktorijuma.

```
    while ( (dirp = readdir(dp)) != NULL)
        printf("%s\n", dirp->d_name);
```

Pokazivač direktorijuma se vraća na početak.

```
    rewinddir(dirp);
```

Zatvaranje direktorijuma.

```
    closedir(dp);
}
```



Funkcije za rad sa direktorijumom

Kreiranje direktorijuma

```
#include <sys/stat.h>  
int mkdir(const char * pathname, mode_t mode);
```

Brisanje direktorijuma

```
#include <sys/stat.h>  
int rmdir(const char * pathname);
```



Funkcije za rad sa direktorijumom

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/file.h>
#include <dirent.h>
#include <sys/stat.h>
#include <sys/types.h>
#include <fcntl.h>

//Maksimalna velicina bafera
#define MAX_BUFF[256]
//Pomocne funkcije
void processdat(char * name);
void processdir(char * name);

int main(int argc, char * argv[])
{
    int result ;
    struct stat statbuf ;
    mode_t mode ;

    //Polazni direktorijum se zadaje kao argument
    if (argc < 2)
    {
        printf("Nema dovoljno argumenata");
        exit(1);
    }
}
```

```
result = stat (argv[1], &statbuf);

//Da li je prosledeno ime direktorijum
if (result == -1)
{
    printf("Ne moze se otvoriti direktorijum");
    exit(1);
}

//Ako je sve u redu direktorijum se obrađuje
mode = statbuf.st_mode;
if (S_ISDIR (mode))
    processdir (argv[1]) ;
}
```



Funkcije za rad sa direktorijumom

```
//Funkcija koja obrađuje sadržaj direktorijuma
void processdir(char * name)
{
    DIR *dp;
    struct dirent *dirp;
    int result ;
    struct stat statbuf ;
    mode_t mode ;

    if ( (dp = opendir(name)) == NULL)
    {
        printf("Greška pri otvaranju direktorijuma");
        exit(1);
    }

    //obrađa stavki iz direktorijuma
    while ( (dirp = readdir(dp)) != NULL)
    {
        //Određuju se atributi objekta iz direktorijuma
        result = stat (dirp->d_name, &statbuf);

        //Došlo je do greške prilikom određivanja atributa
        if (result == -1)
        {
            printf("Greška pri otvaranju direktorijuma");
            exit(1);
        }
    }
}
```

```
mode = statbuf.st_mode;

    //Ime datoteke se prosleđuje na obradu
    if (S_ISREG (mode))
        processdat(dirp->d_name);
    }

    //Zatvaranje direktorijuma
    closedir(dp);
}

//Prikaz datoteke na standardnom izlazu
void processdat(char * name)
{
    int fd;
    char buff[MAX_BUFF];
    int n;

    fd = open(name, O_RDONLY);

    printf("Ime datoteke: %s\n", name);

    while ((n = read(fd, buff, MAX_BUFF)) > 0)
        printf("%s", buff);

    close(fd);
}
```