SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 3103.

OSTVARENJE POJEDNOSTAVLJENOG DATOTEČNOG PODSUSTAVA

Irena Mužar

Zagreb, lipanj 2013.

Sadržaj

1. Uvod

Gotovo se svakodnevno susrećemo s datotekama. One se koriste u radu svakog računalnog sustava, primjerice za razmjenu podataka između programa, za zapis korisničkih podataka i slično.

Datoteku čini skup, prema nekom kriteriju, organiziranih podataka. Kako bismo mogli koristiti podatke zapisane u njoj, potrebno ih je trajno pohraniti. Pošto se sadržaj radnog spremnika gubi nakon svakog isključivanja računala, oni se ne mogu pohraniti u njega. Stoga se datoteke čuvaju na vanjskim spremnicima poput tvrdog diska.

Slika 1.1. Tvrdi disk

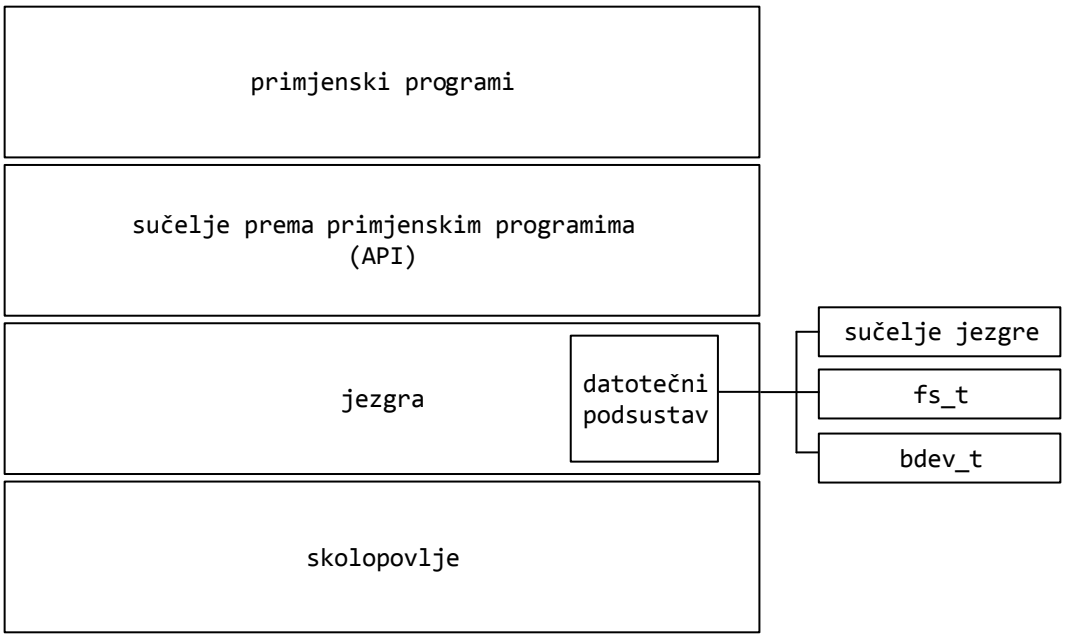
Rad sa datotekama omogućuje datotečni sustav i on je sastavni dio svih modernih operacijskih sustava.

Do danas je razvijeno mnoštvo različitih datotečnih sustava, a nekako su najbolje prihvaćeni datotečni sustavi operacijskih sustava *Windows* i *Linux*. *Windows* operacijski sustavi koriste *FAT* (eng. *File Allocation Table*) i *NTFS* (eng. *New Technology File System*), a *Linux extN* (eng. *extended file system)* datotečne sustave. Svaki od tih sustava različito upravlja datotekama.

Kako bismo opisali jedan od mogućih načina upravljanja datotekama, programski smo ostvarili jedan datotečni sustav, imena *FSE*, što je kratica od eng. *File System Example*. Također, ostvaren je i datotečni podsustav operacijskog sustava koji upravlja tim datotečnim sustavom. Ostvarenje navedenog opisano je u nastavku.

2. Komponente datotečnog podsustava

Računalni sustav (tj. operacijski sustav u proširenom značenju) mora omogućiti izvođenje vrlo složenih operacija. Zbog toga se on ostvaruje na nekoliko razina prikazanih na slici 2.1. Osnovnu razinu čini sklopovlje, a iznad njega se redom nalaze razina jezgre operacijskog sustava, razina sučelja za primjenske programe te razina primjenskih (korisničkih) programa. Operacije i strukture podataka svake razine, skrivene su od drugih razina. Pritom svaka razina nudi sučelje razini koja se nalazi neposredno iznad nje. Na taj se način ostvaruje komunikacija između različitih razina.



Slika 2.1. Razine operacijskog sustava

Datotečni sustav jedan je od osnovnih dijelova jezgre operacijskog sustava. On mora omogućiti upravljanje datotekama koje se nalaze na nekom uređaju.

Ostvarenje koje je prikazano u ovom radu (prema uzoru na prave sustave) dijeli izvedbu datotečnog sustava u nekoliko razina, prema slici 2.1.

Sučelje *bdev_t* omogućuje komunikaciju sa samim uređajem (*bdev_t* koristi upravljačke programe uređaja). Ovisno o tipu datotečnog sustava, operacije nad njime se različito obavljaju. Stoga se u razini označenoj s *fs_t* ostvaruju različiti upravljački programi za različite tipove datotečnog sustava.

Nadalje, datotečni sustav operacijskog sustava mora primjenskim programima ponuditi sučelje pomoću kojih će oni moći koristiti funkcije datotečnog sustava bez poznavanja detalja o upravljanju njime. To smo sučelje nazvali sučelje jezgre.

Uz to, kako bismo pokazali na koji bi način datotečni podsustav operacijskog sustava mogao upravljati datotečnim sustavom na uređaju, osmišljen je i ostvaren datotečni sustav *FSE*. On datotečnom podsustavu operacijskog sustava pruža sučelje *fs_t*.

2.1. Sučelje za korištenje naprava

Datoteke se uobičajeno pohranjuju na vanjske spremnike poput tvrdog diska ili nekih manjih spremnika poput kompaktnog diska. Kako bismo prikazali rad datotečnog sustava, nećemo koristiti neku pravu napravu, nego, radi jednostavnosti, dio spremnika preko sučelja *bdev_t*.

```
struct_bdev_t_;
typedefstruct_bdev_t_bdev_t;

struct_bdev_t_
{
        int     (*init) ( bdev_t *device, size_t size, void *param );
        ssize_t (*read) ( bdev_t *device, size_t offset, size_t size, void *buffer );
        ssize_t (*write) ( bdev_t *device, size_t offset, size_t size, void *buffer );
        ssize_t (*get_size) ( bdev_t *device );
        int     (*callback) ( bdev_t *device );
        size_t    size;
        void     *param;
};
```

Slika 2.2. Sučelje *bdev_t*

U našem ostvarenju, sučeljem je definirano nekoliko jednostavnih funkcija koje će nam omogućiti upravljanje podacima na uređaju. Među njih spadaju funkcija za inicijalizaciju uređaja, funkcija za čitanje podataka sa uređaja te funkcija za pisanje podataka na uređaj.

Funkcija *init()* služi za inicijalizaciju uređaja. Ona kao parametre prima kazaljku na strukturu koja ostvaruje uređaj, veličinu uređaja te kazaljku koja može poslužiti za prijenos nekih podataka. Njena je zadaća inicijalizirati opisnik uređaja. Ako je inicijalizacija bila uspješna, funkcija će vratiti 0. U protivnom treba vratiti -1.

Funkcija *read()* služi za čitanje podataka sa uređaja. Ona kao parametre prima kazaljku na strukturu koja ostvaruje uređaj, poziciju okteta od kojeg se čitaju podaci, broj okteta koji treba pročitati te kazaljku na spremnik u koji se pohranjuju pročitani

podaci. Ako pozicija početnog okteta premašuje veličinu uređaja, neće se pročitati ništa, ali se neće niti javiti greška. Ako pak broj okteta koji se trebaju pročitati premašuje veličinu uređaja, onda će se pročitati najveći mogući broj okteta koji se može pročitati. Povratna vrijednost funkcije je broj okteta koji su uspješno pročitani. U slučaju greške treba vratiti -1.

Na sličan način radi i funkcija *write()* koja služi za pisanje podataka na uređaj. Ona kao parametre prima kazaljku na strukturu koja ostvaruje uređaj, poziciju okteta od kojeg se zapisuju podaci, broj okteta koje treba zapisati te kazaljku na spremnik iz kojeg se podaci zapisuju na uređaj. Ako pozicija početnog okteta premašuje veličinu uređaja, neće se zapisati ništa, ali se neće niti javiti greška. Ako pak broj okteta koji se trebaju zapisati premašuje veličinu uređaja, onda će se zapisati najveći mogući broj okteta koji se može zapisati. Povratna vrijednost funkcije je broj okteta koji su uspješno zapisani. U slučaju greške treba vratiti -1.

Funkcija *get_size()* kao parametar prima kazaljku na strukturu koja ostvaruje uređaj. Ona dohvaća veličinu uređaja, ako će nam ona biti potrebna. Ako se ona ne može dohvatiti, treba vratiti -1.

Također, u sučelju *bdev_t* su definirani parametri *size* i *param*. Prvi označava veličinu uređaja u oktetima, a drugi se koristi za prijenos potrebnih podataka.

Funkcija *callback()* namijenjena je preusmjeravanju izvođenja prema nekoj od jezgrinih funkcija nakon obrade prekida naprave (gdje je to podržano).

U ostvarenju datotečnog sustava *FSE* funkcije sučelja *bdev*_t se mogu koristiti jednostavnije preko njihovih omotača prikazanih na slici 2.3.

```
int bdev_init ( bdev_t *device, size_t size, void *param );

ssize_t bdev_read ( bdev_t *device, size_t offset, size_t size, void *buffer );

ssize_t bdev_write ( bdev_t *device, size_t offset, size_t size, void *buffer );

ssize_t bdev_get_size ( bdev_t *device );
```

Slika 2.3. Omotači za sučelja naprave

2.2. Sučelje za ostvarenje datotečnog sustava

Datotečnom sustavu *FSE* iz datotečnog podsustava se pristupa preko sučelja *fs_t* prikazanom na slici 2.4. Njime su definirane, u našem ostvarenju, osnovne funkcije za rad sa datotečnim sustavom. Tu spadaju funkcije za inicijalizaciju praznog uređaja, funkcije za otvaranje i zatvaranje datotečnog sustava te funkcije za rad sa datotekama. Njih čine funkcije za otvaranje i zatvaranje datoteke, čitanje iz datoteke, pisanje u datoteku te pozicioniranje unutar datoteke.

```
struct_fs_t_;
typedefstruct_fs_t_fs_t;

struct_fs_t_
{
        int (*init_fs) (fs_t *fs, size_t bsize, size_t bcount, bdev_t *device);
        int (*open_fs) ( fs_t *fs, bdev_t *device );
        int (*close_fs) ( fs_t *fs );
        fs_node_t *(*open) ( fs_t *fs, char *, int flags );
        fs_node_t *(*open_by_id) ( fs_t *fs, fs_node_id_t, int flags );
        int (*read) ( fs_t *fs, fs_node_t *fp, void *buffer, size_t size );
        int (*write) ( fs_t *fs, fs_node_t *fp, void *buffer, size_t size );
        ssize_t (*seek) ( fs_t *fs, fs_node_t *fp, ssize_t offset, int whence );
        int (*close) ( fs_t *fs, fs_node_t *fp );
        fsdesc_t fsdesc;
        void *params;
};
```

Slika 2.4. Sučelje *fs_t*

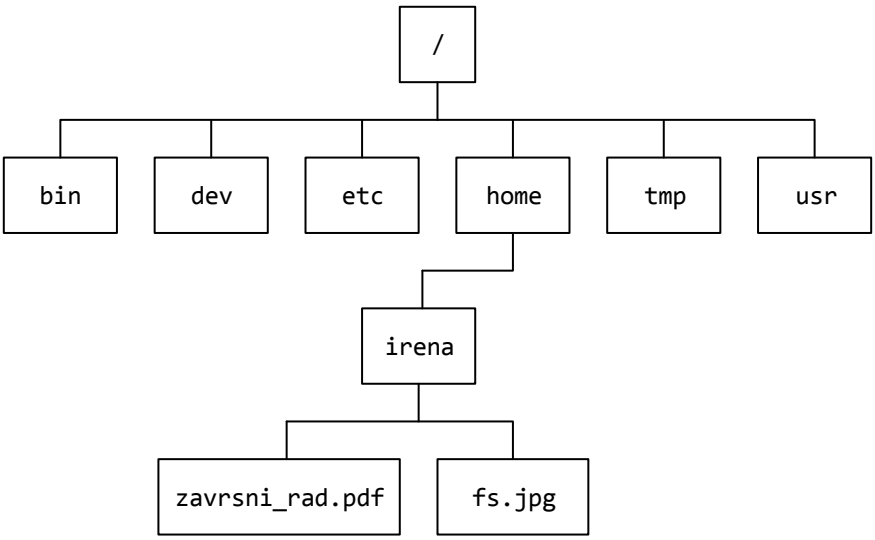
Sučeljem *fs_t* su definirani dodatni parametri *fsdesc* i *params*. Prvi od njih predstavlja opisnik datotečnog sustava koji koristi operacijski sustav, a drugi kazaljku koja može poslužiti za prijenos raznih podataka.

Kako bismo znali na koji način datotečni podsustav ostvaruje upravljanje datotečnim sustavom, opisati ćemo najprije strukture podataka koje koristi datotečni sustav *FSE*.

2.2.1. Strukture podataka datotečnog sustava *FSE*

Uobičajene strukture podataka svakog datotečnog sustava su opisnik datotečnog sustava te opisnici datoteka. Također, željeli bismo omogućiti njihovo organiziranje prema nekom kriteriju. To može biti, primjerice, prema njihovom sadržaju, namjeni, autoru i slično. Za to će nam poslužiti struktura datotečnog sustava koja se naziva direktorijsko stablo (eng. *directory tree*).

Direktorijsko stablo je struktura čiji su čvorovi direktoriji, a listovi datoteke. Pritom su direktoriji logičke jedinice i služe isključivo za organizaciju datoteka, a datoteke sadrže podatke koje je nužno pohraniti negdje na uređaju (npr. prema slici 2.5).



Slika 2.5. Primjer direktorijskog stabla

Svaki uređaj podijeljen je na manje dijelove jednakih veličina koji se nazivaju sektorima. Njih čine nizovi okteta. Međutim, uobičajeno je da se skupina uzastopnih sektora grupira u veću jedinicu podataka koju nazivamo blokovima. Stoga je jedinica podataka sa kojom radi datotečni sustav blok. Pritom kažemo da se datoteci dodijeljuju blokovi.

2.2.1.1. Opisnik datotečnog sustava

Opisnik datotečnog sustava sadrži osnovne podatke o datotečnom sustavu koji se nalazi na uređaju. Za *FSE* ti su podaci prikazani na slici 2.6.

```
typedefstruct_fse_t_
{
        fs_id_t id;
        char label[FSE_LABEL_LEN];
        int  type;
        int  blk_sz;
        int  blk_num;
        int  TFD;
        fse_chunk_t free[FSE_MAX_CHUNKS];
        fse_chunk_t unusable[FSE_MAX_UNUSABLE_CHUNKS];
}
fse_t;
```

Slika 2.6. Opisnik datotečnog sustava *fse_t*

Parametar *id* predstavlja identifikator datotečnog sustava. Parametar *label* predstavlja ime datotečnog sustava, a parametar *type* tip datotečnog sustava. Ove podatke koristi datotečni sustav operacijskog sustava kako bi mogao prepoznati o kojem se datotečnom sustavu radi i upravljati podacima na njemu.

Slijede ih podaci koji će omogućiti opisivanje spremničkog prostora na uređaju. Prvi od njih su parametari *blk_sz* i *blk_num*. Oni označavaju kolika je veličina bloka na uređaju te koliko takvih blokova na njemu postoji.

Da bismo znali kako su blokovi smješteni na uređaju, uvodimo novu strukturu podataka *fse_chunk_t*. Ona opisuje smještaj skupine blokova, pri čemu parametar *ind_fs* predstavlja broj prvog bloka na uređaju, a parametar *cnt* koliko uzastopnih blokova slijedi nakon prvog bloka. Parametar *ind_file* ima dvojako značenje. Ako se radi o datoteci, onda predstavlja logički broj prvog bloka datoteke koji je smješten na navedenom mjestu na uređaju. U protivnom se on ne koristi.

```
typedefstruct_fse_chunk_t_
{
        uint32 ind_file;
        uint32 ind_fs;
        uint32 cnt;
}
fse_chunk_t;
```

Slika 2.7. Struktura *fse_chunk_t*

Struktura *fse_chunk_t* koristi se za opis slobodnih i neupotrebljivih blokova na disku. Slobodni su oni blokovi koje datotečni sustav ne koristi za pohranu podataka i koji se mogu dodijeliti. Neupotrebljivi blokovi su oni blokovi koje datotečni sustav ne može i ne smije dodjeljivati. Razloga tome može biti više, a jedan od njih je oštećenje sektora na uređaju. U opisniku datotečnog sustava *fse_t*, parametar *free* predstavlja slobodne blokove, a parametar *unusable* neupotrebljive blokove.

Primjetimo da smo preskočili parametar *TFD*. On se koristi kako bismo znali gdje su smješteni opisnici datoteka i direktorija. Točnije, označava broj bloka na uređaju na kojem je smješten prvi blok tablice opisnika datoteka i direktorija. Kako se datoteke i direktoriji mogu nazvati čvorovima datotečnog sustava, češće koristimo naziv tablica opisnika čvorova.

2.2.1.2. Opisnici čvorova datotečnog sustava

Opisnici čvorova datotečnog sustava *FSE* ostvareni su strukturom podataka *fse_node_t*.

```
typedefstruct_fse_node_t_
{
        char name[FSE_NAME_MAX_LEN];
        fse_nodeid_t node_id;
        int type;
        fse_node_id_t parent_id;
        timespec_t ctime;
        timespec_t mtime;
        timespec_t atime;
        mode_t mode;
        size_t size;
        union
      {
                fse_chunk_t chunk[FSE_MAX_CHUNKS];
                fse_node_id_t child[FSE_MAX_NODE_IN_DIR];
        };
}
fse_node_t;
```

Slika 2.8. Opisnik čvora datotečnog sustava *fse_node_t*

Parametar *name* predstavlja ime čvora. Ono može sadržavati sve znakove, osim osim znaka "/". On ima posebnu ulogu, koja će biti objašnjena u daljnjim poglavljima. Parametar *node_id* služi kao identifikator čvora u tablici opisnika čvorova. Predstavlja redni broj zapisa u tablici opisnika čvorova. Parametar *type* određuje tip čvora kojeg opisuje opisnik. Tip čvora može biti *FSE_FILE*, *FSE_DIR*, *FSE_LINK*, *FSE_TFD* i *FSE_LAST*. *FSE_FILE* označuje da opisnik čvora opisuje datoteku, *FSE_DIR* direktorij, a *FSE_LINK* poveznicu na drugi datotečni sustav. *FSE_TFD* predstavlja opisnik tablice opisnika čvorova, a *FSE_LAST* se koristi isključivo kao oznaka kraja tablice.

Također, važno je i pohraniti podatke o stvaranju čvora, izmjeni čvora te pristupanju čvora. Za to se koriste parametri *ctime*, *mtime* i *atime*. Parametar *mode* predstavlja dozvolu pristupa čvoru.

Nadalje, dva su podatka vrlo važna za ostvarenje stablaste strukture datotečnog sustava. To su *parent_id* i *child*, odnosno čvoru nadređeni čvor i podređeni čvorovi. Oni su ostvareni strukturom podataka *fse_node_id_t*.

```
typedefstruct_fse_node_id_t_
{
        fse_id_t      fse_id;
        fse_nodeid_t  node_id;
}
fse_node_id_t;
```

Slika 2.9. Struktura *fse_node_id_t*

Parametar *fse_id* strukture *fse_node_id_t* predstavlja identifikator datotečnog sustava na kojem se nalazi čvor. Ako je on jednak nuli, označuje da se čvor nalazi na istom datotečnom sustavu kao i njemu podređeni, odnosno nadređeni čvor. U protivnom označuje da se on nalazi na nekom drugom datotečnom sustavu. Parametar *node_id* predstavlja redni broj zapisa čvora u tablici opisnika čvorova.

Čvor može imati podređene čvorove, samo ako se radi o direktoriju. Stoga se smještaj blokova datoteke u opisniku čvora *fse_node_t* opisuje strukturom podataka *fse_chunk_t*, na način kako je navedeno pri opisu smještaja slobodnih i neupotrebljivih blokova u prethodnom poglavlju. Smještaj blokova datoteke je u strukturi *fse*_t predstavljen parametrom *chunk.*

Uz to, kako bismo mogli čitati podatke iz datoteke ili ih zapisivati u nju, potrebno je znati njenu veličinu. Ona se pohranjuje pomoću parametra *size*.

2.3. Datotečni podsustav operacijskog sustava

Funkcije za upravljanje datotečnim sustavom, kako je definirano sučeljem *fs_t*, da bi bilo razumljivije opisat ćemo u sklopu ovog poglavlja na ostvarenjem za *FSE*. Pritom ćemo najprije opisati strukture podataka koje koristi operacijski sustav, a zatim i same funkcije onim redom kojim su navedeni.

2.3.1. Strukture podataka datotečnog podsustava

Kako bismo znali koji se podaci nalaze na uređaju, potrebno je dio tih podataka učitati u strukture operacijskog sustava. Te su strukture vrlo slične već ranije opisanim strukturama podataka, no strukture podataka operacijskog sustava su generičke, tj. služe tome kako bi on mogao upravljati s više različitih datotečnih sustava.

Opisnik datotečnog sustava koji koristi operacijski sustav ostvaren je strukturom podataka *fsdesc_t*, prema slici 2.10.

```
typedef struct_fsdesc_t_
{
        fs_id_t  id;
        char    *label;
        int      type;
        fs_t    *fs;
        int      refcnt;
}
fsdesc_t;
```

Slika 2.10. Opisnik datotečnog sustava u operacijskom sustavu *fsdesc_t*

Parametar *id* predstavlja redni broj datotečnog sustava koji je otvoren u operacijskom sustavu. Parametar *label* predstavlja ime datotečnog sustava, parametar *type* njegov tip, a parametar *fs* kazaljku na sučelje koje ostvaruje taj datotečni sustav.

Opisnik čvora datotečnog sustava koji koristi operacijski sustav je, ustvari, opisnik datoteke. On je ostvaren strukturom podataka *fs_node_t*, prema slici 2.11.

```
typedefstruct_fs_node_t_
{
        int           des_id;
        fs_node_id_t  node_id;
        char         *node_name;
        void         *descr;
        int           type;
        int           fp;
        int           oflag;
        mode_t        mode;
}
fs_node_t;
```

Slika 2.11. Opisnik datoteke u operacijskom sustavu *fs_node_t*

Parametar *des_id* predstavlja redni broj datoteke koja je otvorena u datotečnom sustavu. Zatim, parametar *node_id* predstavlja identifikaciju čvora. On je opisan strukturom podataka vrlo sličnoj strukturi *fse_node_id_t*, kojom je ostvaren prikaz podređenog i nadređenog čvora u prethodnom poglavlju. Međutim, postoji razlika. Ovdje prvi parametar predstavlja identifikator otvorenog datotečnog sustava.

Ime datoteke predstavlja *node_name* i zadaje se apsolutnom putanjom. Parametar *descr* je kazaljka na kopiju opisnika datoteke sa uređaja u memoriji. Parametar *type* predstavlja tip čvora, a *mode* dozvolu pristupa.

Kako bismo mogli ostvariti operacije čitanja i pisanja, potrebna nam je i datotečna kazaljka koja će nam reći, gdje je operacijski sustav stao sa radom. Naravno, neke će datoteke biti stvorene samo za čitanje, druge samo za pisanje te je zato potrebno i

uvesti zastavicu s kojom ćemo označiti način rada sa datotekom. Ti su podaci pohranjeni pomoću parametara *fp* i *oflag*.

2.3.2. Funkcije za rad datotečnog podsustava

Na slici 2.4. prikazano je sučelje *fs_t* kojim su definirane funkcije za inicijalizaciju praznog datotečnog sustava na uređaju, otvaranje i zatvaranje datotečnog sustava te za rad s datotekama.

2.3.2.1. Inicijalizacija datotečnog sustava na uređaju

Funkcija *init_fs()* služi za inicijalizaciju datotečnog sustava na uređaju. Njena je namjena zapisati osnovne podatke o datotečnom sustavu i datotekama na uređaj, kako bi on bio spreman za uporabu od strane datotečnog podsustava. Stoga u prve blokove uređaja moramo zapisati opisnik datotečnog sustava te tablicu opisnika čvorova.

Prije zapisivanja, opisnici se moraju inicijalizirati. Početno stanje opisnika datotečnog sustava prikazano je na slici 2.12.

```
fse_t:
        id = 0
        label = FSE-example
        type = 1
        block_size = 512
        block_count = 128
        TFD = 1
        free[]:
                { 0, 6, 122 }
                { 0, 0, 0 }
                ...
                { 0, 0, 0 }
        unusable[]:
                { 0, 0, 0 }
                ...
                { 0, 0, 0 }
```

Slika 2.12. Početno stanje opisnika datotečnog sustava

Nakon inicijalizacije opisnik se pohranjuje u prvi blok uređaja te se potom od prvog slobodnog bloka na uređaju zapisuje tablica opisnika čvorova.

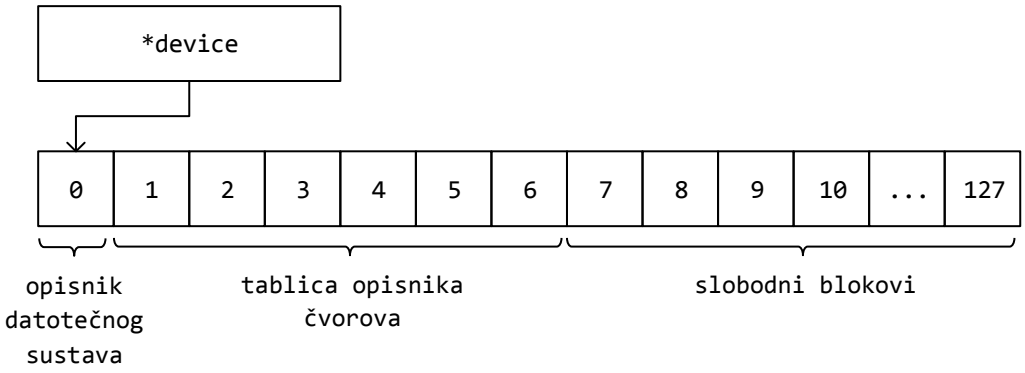
Tablica opisnika čvorova sadrži skup opisnika. Kako bismo si olakšali snalaženje u njoj, dva će opisnika imati posebnu namjenu. Tako će prvi opisnik biti opisnik same tablice opisnika čvorova, a posljednji će se samo koristiti kako bi označio kraj tablice.

U početku će svi preostali opisnici biti prazni, a kasnije će se koristiti za opis nekih čvorova. Pošto opisnici sadrže veliku količinu podataka, na slici je prikazano samo početno stanje opisnika tablice čvorova. Analogno se popunjavaju preostali opisnici.

```
fse_node_t:
        node_id = 0
        name = TFD
        type = 1
        parent_id:
                [fse_node_id_t]={ 0, 0 }
        chunks:
                [fse_chunk_t]={ 0, 1, 5 }
                [fse_chunk_t]={ 0, 0, 0 }
                ...
                [fse_chunk_t]={ 0, 0, 0 }
```

Slika 2.13. Početno stanje opisnika tablice opisnika čvorova

Funkcija koja obavlja navedeno – *init_fs()*, kao parametre prima kazaljku na sučelje koje ostvaruje datotečni sustav, veličinu blokova na uređaju, broj blokova na uređaju te kazaljku na uređaj na koji se zapisuju opisnici. Ako je inicijalizacija provedena na ispravan način, funkcija treba vratiti 0. U protivnom treba vratiti -1. Nakon inicijalizacije, uređaj bi trebao izgledati kao na slici 2.15.



Slika 2.15. Stanje datotečnog sustava nakon inicijalizacije

2.3.2.2. Otvaranje datotečnog sustava

Učitavanje podataka o datotečnom sustavu i datotekama koje se nalaze na njemu u strukture podataka operacijskog sustava naziva se otvaranje datotečnog sustava. Ono podrazumijeva učitavanje opisnika datotečnog sustava i zapisivanje podataka u generički opisnik datotečnog sustava. Taj će se opisnik u daljnjem radu nazivati

opisnik otvorenog datotečnog sustava. Datotečni podsustav operacijskog sustava može istovremeno upravljati podacima sa više uređaja. Kako bismo mogli upravljati datotekama koje se nalaze na uređaju, potrebno je i učitati cijelu tablicu opisnika u memoriju. U tome će nam uvelike pomoći, ako definiramo jednu dodatnu strukturu podataka, koju ćemo nazvati *fse_rt_t*.

```
typedefstruct_fse_rt_t_
{
        fse_t       *descr;
        bdev_t      *device;
        fse_node_t  *tfd;
}
fse_rt_t;
```

Slika 2.16. Struktura *fse_rt_t*

Ona sadrži kazaljku na kopiju opisnika datotečnog sustava u memoriji, kazaljku na sam uređaj te kazaljku na kopiju tablice opisnika čvorova datotečnog sustava u memoriji.

Funkcija za otvaranje datotečnog sustava, *open_fs()*, kao parametre prima kazaljku na sučelje koje ostvaruje datotečni sustav te kazaljku na uređaj koji treba otvoriti. Njena je zadaća izvršiti opisano. Tablica opisnika, kako bi se mogla koristiti u nastavku, prenosi se preko sučelja. Ako je otvaranje provedeno na ispravan način, funkcija treba vratiti 0. U protivnom treba vratiti -1.

2.3.2.3. Zatvaranje datotečnog sustava

Zatvaranje datotečnog sustava podrazumijeva zatvaranje datoteka koje se nalaze na uređaju, zapisivanje opisnika datotečnog sustava i tablice opisnika čvorova na uređaj te oslobađanje memorijskog prostora koje je datotečni podsustav koristio.

Funkcija za zatvaranje datotečnog sustava, *close_fs()*, kao parametar prima kazaljku na sučelje koje ostvaruje datotečni sustav. Njena je zadaća izvršiti opisano. Ako je zatvaranje provedeno na ispravan način, funkcija treba vratiti 0. U protivnom treba vratiti -1.

2.3.2.4. Otvaranje datoteke

Kako bismo mogli čitati podatke iz datoteke ili pisati u nju, datoteku je prvo potrebno otvoriti. Otvaranje datoteke podrazumijeva dohvaćanje kopije opisnika datoteke iz

memorije i njegovo zapisivanje u opisnik datoteke operacijskog sustava. Taj se opisnik u daljnjem radu naziva *opisnik otvorene datoteke*. Funkcija za otvaranje datoteke, *open()*, kao parametre prima kazaljku na sučelje koje ostvaruje datotečni sustav, kazaljku na puno ime datoteke te zastavice uz koje je datoteka otvorena. Povratna vrijednost funkcije je kazaljka na otvoreni opisnik datoteke, a u slučaju greške treba vratiti *NULL*.

Ime datoteke mora biti zadano apsolutno, što znači da će ono otkrivati podatke o točnom smještaju datoteke unutar datotečnog sustava. Sastoji se od niza imena odvojenih znakom "/", pri čemu posljednji član predstavlja ime datoteke, a preostali članovi su direktoriji. Također, znak "/" koristi se i kao oznaka korijenskog direktorija i mora se navesti na početku apsolutnog imena svake datoteke. Svi preostali znakovi mogu se slobodno koristiti u imenu. Uz to, valja reći da se ekstenzijama ne pridaje značenje, već je to zadaća viših slojeva. Primjer jednog takvog imena može biti "/dir1/file11.x".

Svaka se datoteka može otvoriti na više različitih načina. Ti su načini određeni zastavicama *O_RDONLY*, *O_WRONLY*, *O_RDWR*, *O_APPEND* i *O_CREAT*. Zastavica *O_RDONLY* označuje da je datoteka otvorena samo za čitanje, *O_WRONLY* samo za pisanje, a *O_RDWR* i za čitanje i pisanje. Bilo koja kombinacija prethodnih, nije dozvoljena. Zastavica *O_APPEND* zadaje se ili uz zastavicu *O_WRONLY* ili *O_RDWR*. Time se omogućuje pisanje u datoteku, počevši od njenog kraja. Zastavica *O_CREAT*, također, se zadaje uz zastavice *O_WRONLY* ili *O_RDWR*, pri čemu se omogućuje stvaranje datoteke.

Ako je postavljena kombinacija zastavica koja sadrži *O_CREAT*, funkcija će stvoriti sve potrebne čvorove i zapisati ih u tablicu opisnika u memoriji. Napomenimo još, da se zastavice *O_APPEND* i *O_CREAT* nužno moraju postaviti uz prethodno navedene zastavice.

Ako je postavljena zastavica *O_WRONLY* nad već postojećom datotekom, funkcija će otpustiti sve blokove koji su joj bili dodijeljeni.

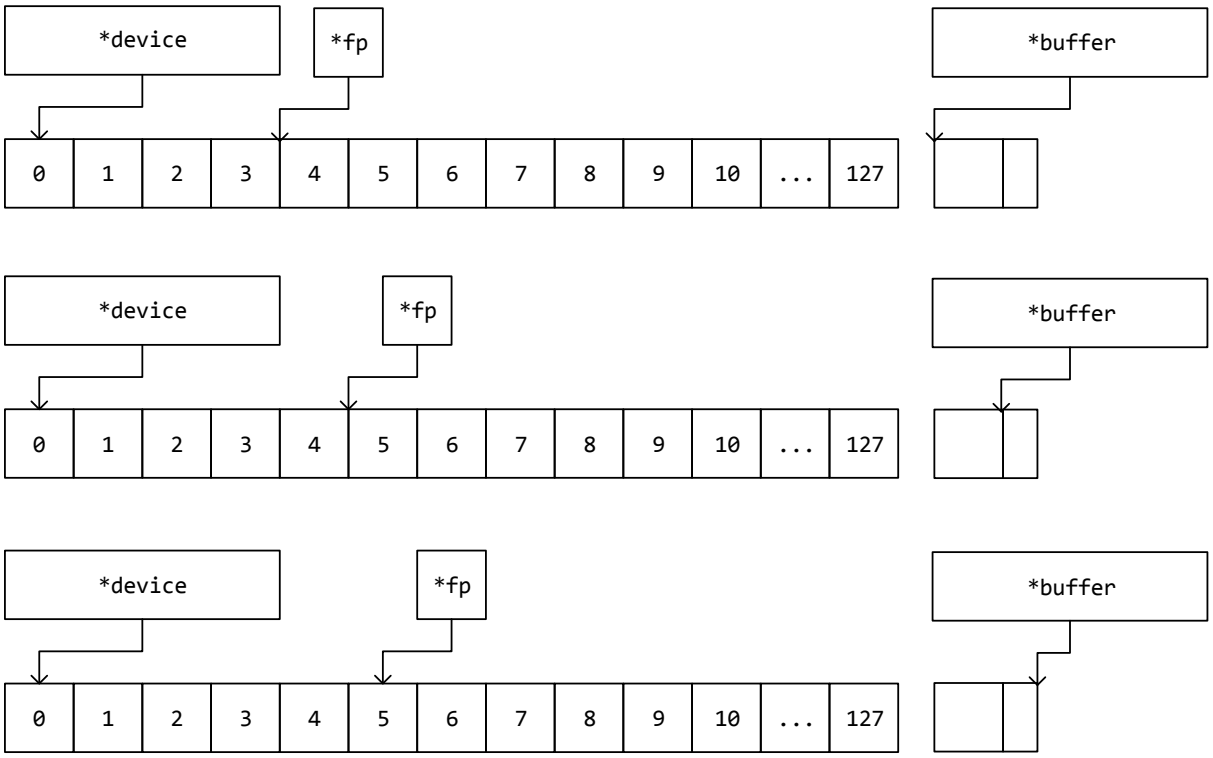
2.3.2.5. Čitanje iz datoteke

Čitanje iz datoteke podrazumijeva dohvat zadanog broja podataka datoteke sa uređaja, te njihovo zapisivanje u pomoćni spremnik. Kako bismo to mogli napraviti,

prije svega, datoteku je potrebno otvoriti uz zastavicu koja omogućuje čitanje iz datoteke, kako je navedeno pri opisu funkcije za otvaranje datoteke.

Podaci se sa uređaja čitaju od pozicije datotečne kazaljke pa sve do pozicije datotečne kazaljke uvećane za broj okteta koje je potrebno pročitati. Ako se ne može pročitati zadani broj okteta, on će se smanjiti na broj okteta koje je moguće pročitati.

Kako podaci datoteke mogu biti smješteni bilo gdje na uređaju, oni se čitaju blok po blok. Prvo se čitaju podaci iz bloka na kojeg pokazuje datotečna kazaljka, a zatim i blokovi u kojima se nalaze preostali podaci datoteke. Kako se podaci čitaju, tako se pomiče i datotečna kazaljka. Primjer čitanja podataka iz datoteke prikazan je na slici 2.17.



Slika 2.17. Primjer čitanja podataka iz datoteke

Primjer prikazuje postupak čitanja podataka iz datoteke koja sadrži podatke zapisane u četvrtom i petom bloku na uređaju.

Funkcija za čitanje iz datoteke, *read()*, kao parametre prima kazaljku na sučelje koje ostvaruje datotečni sustav, kazaljku na opisnik otvorene datoteke, kazaljku na spremnik u koji treba zapisati podatke pročitane s uređaja te broj okteta koje je

potrebno pročitati. Povratna vrijednost funkcije je broj stvarno pročitanih okteta, a u slučaju greške treba vratiti -1.

2.3.2.6. Pisanje u datoteku

Pisanje u datoteku podrazumijeva zapisivanje zadanog broja podataka iz pomoćnog spremnika u blokove datoteke na uređaju. Prije pisanja, datoteku je potrebno otvoriti uz zastavicu koja omogućuje pisanje u datoteku, kako je navedeno pri opisu funkcije za otvaranje datoteke.

Ovisno o načinu na koji je datoteka otvorena i poziciji datotečne kazaljke, datoteci se dodjeljuju blokovi potrebni za zapis podataka na uređaj. Funkcija nastoji dodijeliti skup slobodnih blokova (potrebnih za pisanje):

1. smještenih neposredno nakon zadnjeg bloka datoteke, ako se radi o postojećoj datoteci
2. smještenih bilo gdje.

Korak 2 ponavlja se sve dok se ne dodijele svi potrebni blokovi (ako svi potrebni nisu dodijeljeni u prvom koraku).

Podaci se na uređaj zapisuju od pozicije datotečne kazaljke pa sve do pozicije datotečne kazaljke uvećane za broj okteta koje je potrebno zapisati. Ako se ne može zapisati zadani broj okteta, on će se smanjiti na broj okteta koje je moguće zapisati.

Kako podaci datoteke mogu biti smješteni bilo gdje na uređaju, oni se zapisuju blok po blok. Prvo se zapisuju podaci u blok na kojem se nalazi datotečna kazaljka, a zatim i preostale slobodne blokove. Kako se podaci zapisuju, tako se pomiče i datotečna kazaljka. Pogledamo li ponovno sliku 2.17. iz prethodnog poglavlja, uočit ćemo da je čitanje vrlo slično postupku pisanja podataka u datoteku.

Funkcija za pisanje u datoteku, *write()*, kao parametre prima kazaljku na sučelje koje ostvaruje datotečni sustav, kazaljku na opisnik otvorene datoteke, kazaljku na spremnik iz kojeg treba pročitati podatke koje treba zapisati na uređaj te broj okteta koje je potrebno zapisati. Povratna vrijednost funkcije je broj stvarno zapisanih okteta, a u slučaju greške treba vratiti -1.

2.3.2.7. Pozicioniranje unutar datoteke

Pozicioniranje unutar datoteke podrazumijeva promjenu položaja datotečne kazaljke.

Funkcija za pozicioniranje, *seek()*, kao parametre prima kazaljku na sučelje koje ostvaruje datotečni sustav, kazaljku na opisnik otvorene datoteke, pomak od položaja datotečne kazaljke te zastavicu koja postavlja datotečnu kazaljku. Povratna vrijednost funkcije je pozicija datotečne kazaljke nakon promjene, a u slučaju greške treba vratiti -1.

Zastavica može poprimiti vrijednost *SEEK_SET*, *SEEK_CUR* ili *SEEK_END*. Zastavica *SEEK_SET* postavlja datotečnu kazaljku na poziciju udaljenu za pomak okteta od početka datoteke. Zastavica *SEEK_CUR* postavlja datotečnu kazaljku na poziciju udaljenu za pomak okteta u odnosu na trenutnu poziciju kazaljke. Zastavica *SEEK_END* postavlja datotečnu kazaljku udaljenu za pomak okteta od kraja datoteke. Važno je naglasiti da pomak može biti negativan. Tada će se datotečna kazaljka pomicati unatrag.

2.3.2.8. Zatvaranje datoteke

Zatvaranje datoteke podrazumijeva zapisivanje potrebnih opisnika na uređaj te oslobađanje memorije koju zauzima opisnik otvorene datoteke.

Funkcija za zatvaranje datoteke, *close()*, kao parametre prima kazaljku na sučelje koje ostvaruje datotečni sustav te kazaljku na opisnik otvorene datoteke. Povratna vrijednost funkcije je 0, a u slučaju greške treba vratiti -1.

2.4. Sučelje jezgre za rad s datotečnim podsustavom

Sučelje za koje datotečni podsustav nudi višim slojevima prikazano je na slici 2.18.

```
int kfs_init ();
int kfs_add_mount_point ( char *path, fs_id_t fs_id );
int kfs_remove_mount_point ( char *path );
int kfs_remove_mount_point_by_id ( int id );
int kfs_find_mount_point ( char *path );
int kfs_init_fs ( fs_t *fs, size_t bsize, size_t bcount, bdev_t *device );
fs_id_t kfs_open_fs ( fs_t *fs, bdev_t *device );
int kfs_close_fs ( fs_id_t fs_id );
fs_node_t *kfs_open ( char *file_name, int flags );
int kfs_close ( fs_node_t *fp );
fs_node_t *kfs_open_by_name ( fsdesc_t *fsdes, char *file_name, int flags );
fs_node_t *kfs_open_by_id ( fs_node_id_t node, int flags );
int kfs_read ( fs_node_t *fp, void *buffer, size_t size );
int kfs_write ( fs_node_t *fp, void *buffer, size_t size );
ssize_t kfs_seek ( fs_node_t *fp, ssize_t offset, int whence );
```

Slika 2.18. Sučelje datotečnog podsustava prema višim slojevima

Opis rada većine navedenih funkcija već smo objasnili, međutim, postoje i funkcije čiji rad još moramo opisati. To su funkcije za priključivanje datotečnog sustava.

Naime, datotečni podsustav operacijskog sustava mora omogućiti rad sa više uređaja. Kako bi se uređaj mogao upotrijebiti, prije njegove uporabe, potrebno ga je priključiti u već otvoreni datotečni sustav. Mjesto na koje se on priključuje je, ustvari, jedan od direktorija i naziva se točka priključka, a sam se postupak naziva priključivanje (eng. mount).

Točka priključka pritom postaje korijenski direktorij novog datotečnog sustava. Ako je u direktoriju postojao neki drugi zapis, on postaje nevidljiv za datotečni sustav operacijskog sustava sve dok se novi datotečni sustav ne odspoji, odnosno dok se ne ukloni točka priključka.

Struktura podataka koja se koristi za opis točke priključka je *mount_point_t*.

```
typedefstruct_mount_point_t_
{
        char    *path;
        fs_id_t  fs_id;
}
mount_point_t;
```

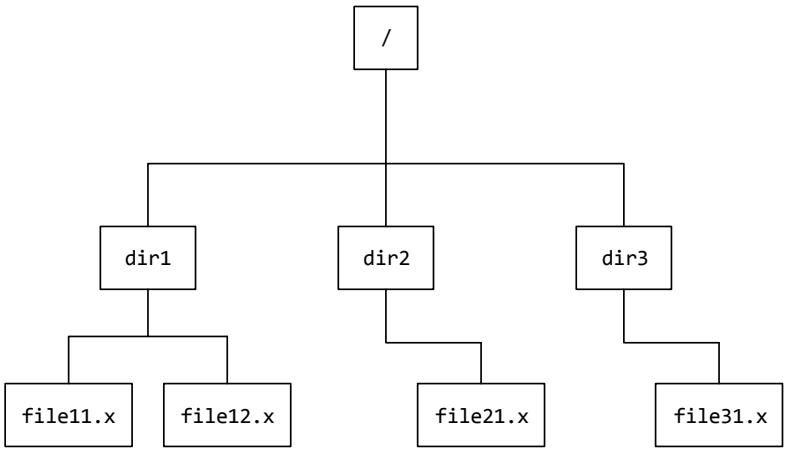
Slika 2.19. Struktura *mount_point_t*

Parametar *path* predstavlja kazaljku na apsolutno ime direktorija na koji se datotečni sustav priključuje, a parametar *fs_id* predstavlja identifikator otvorenog datotečnog sustava koji se na njega priključuje.

Funkcije koje upravljaju priključivanjem datotečnih sustava su *kfs_add_mount_point()* i *kfs_remove_mount_point()*. Funkcije u slučaju ispravnog izvršavanja trebaju vratiti 0, a u protivnom -1.

3. Primjer rada datotečnog podsustava

Na jednostavnom primjeru datotečnog sustava (slika 3.1.) prikazuju se strukture podataka datotečnog podsustava. Redom su prikazani opisnik otvorenog datotečnog sustava, opisnik datotečnog sustava, tablica opisnika čvorova te opisnici otvorenih datoteka.



Slika 3.1. Primjer datotečnog sustava na uređaju

Opisnici su prikazani onako, kako izgledaju nakon otvaranja datotečnog sustava te otvaranja datoteka "file11.x" i "file31.x".

Napomenimo da su imena datoteka i direktorija namjerno numerirana, kako bismo naglasili način na koji je organiziran datotečni sustav.

3.1. Opisnik otvorenog datotečnog sustava

U datotečnom podsustavu operacijskog sustava otvoren je samo jedan datotečni sustav.

```
fsdesc_t[]:
        id = 0
        label = FSE-example
        type = 1
        fs = 0x00123456
        refcnt = 1
```

Slika 3.2. Opisnik otvorenog datotečnog sustava

Podsjetimo se, parametar *fs* u opisniku je kazaljka na sučelje koji ostvaruje datotečni sustav *FSE*. U našem ostvarenju, preko tog se sučelja, pomoću kazaljke *params*, prenose podaci koji su učitani prilikom otvaranja datotečnog sustava. To su opisnik datotečnog sustava te tablica opisnika čvorova.

3.1.1. Opisnik datotečnog sustava

```
fse_t:
        id = 0
        label = FSE-example
        type = 1
        block_size = 512
        block_count = 128
        TFD = 1
        free[]:
                { 0, 10, 118 }
```

Slika 3.3. Opisnik datotečnog sustava

3.1.2. Tablica opisnika čvorova datotečnog sustava

U spremnički prostor učitana je cijela tablica opisnika čvorova. Svaki opisnik tablice prikazan je posebno.

```
fse_node_t:
        node_id = 0
        name = TFD
        type = 1
        parent_id:
                [fse_node_id_t]={ 0, 0 }
        chunks:
                [fse_chunk_t]={ 0, 1, 5 }
```

```
fse_node_t:
        node_id = 1
        name = /
        type = 2
        parent_id:
                [fse_node_id_t]={ 0, -1 }
        child[]:
                [fse_node_id_t]={ 0, 2 }
                [fse_node_id_t]={ 0, 5 }
                [fse_node_id_t]={ 0, 7 }
```

```
fse_node_t:
        node_id = 2
        name = dir1
        type = 2
        parent_id:
                [fse_node_id_t]={ 0, 1 }
        child[]:
                [fse_node_id_t]={ 0, 3 }
                [fse_node_id_t]={ 0, 4 }
```

```
fse_node_t:
        node_id = 3
        name = file11.x
        type = 3
        parent_id:
                [fse_node_id_t]={ 0, 2 }
        size = 500
        chunks:
                [fse_chunk_t]={ 0, 6, 1 }
```

```
fse_node_t:
        node_id = 4
        name = file12.x
        type = 3
        parent_id:
                [fse_node_id_t]={ 0, 2 }
        size = 321
        chunks:
                [fse_chunk_t]={ 0, 7, 1 }
```

```
fse_node_t:
        node_id = 5
        name = dir2
        type = 2
        parent_id:
                [fse_node_id_t]={ 0, 1 }
        child[]:
                [fse_node_id_t]={ 0, 6 }
```

```
fse_node_t:
        node_id = 6
        name = file21.x
        type = 3
        parent_id:
                [fse_node_id_t]={ 0, 5 }
        size = 428
        chunks:
                [fse_chunk_t]={ 0, 8, 1 }
```

```
fse_node_t:
        node_id = 7
        name = dir3
        type = 2
        parent_id:
                [fse_node_id_t]={ 0, 1 }
        child[]:
                [fse_node_id_t]={ 0, 8 }
```

```
fse_node_t:
        node_id = 8
        name = file31.x
        type = 3
        parent_id:
                [fse_node_id_t]={ 0, 7 }
        size = 274
        chunks:
                [fse_chunk_t]={ 0, 9, 1 }
```

```
fse_node_t:
        type = -2
```

Slika 3.4. Tablica opisnika čvorova datotečnog sustava

3.2. Opisnici otvorenih datoteka

```
fs_node_t:
        des_id = 0
        node_id:
                [fs_node_id_t]={ 0, 3 }
        node_name = /dir1/file11.x
        type = 0
        fp = 0
        oflag = 2
        mode = 0
```

```
fs_node_t:
        des_id = 1
        node_id:
                [fs_node_id_t]={ 0, 8 }
        node_name = /dir3/file31.x
        type = 0
        fp = 0
        oflag = 2
        mode = 0
```

Slika 3.5. Opisnici otvorenih datoteka

Napomenimo, također, da nisu svi prikazani opisnici potpuni. Neke smo podatke izostavili jer se ili ne koriste ili nisu bitni za prikaz osnovne ideje rada datotečnog podsustava.

4. Zaključak

U ovom je radu ostvaren i objašnjen pojednostavljeni datotečni podsustav koji se može ugraditi u operacijski sustav za ugradbena računala. Njegovo ostvarenje sadrži osnovne strukture podataka i operacije koje svaki datotečni sustav mora sadržavati. To uključuje otvaranje i zatvaranje datotečnog sustava te osnovne operacije za rad sa datotekama: otvaranje i zatvaranje, čitanje i pisanje te pozicioniranje unutar datoteke.

Rad je osmišljen na način da ostvarenje programskog produkta prikazuje onako, kako je u stvarnosti tekao njegov razvoj. Najprije je ostvareno sučelje *bdev*_t koje definira osnovne operacije za rad s uređajem. Nakon toga osmišljen je i ostvaren datotečni sustav *FSE*, koji će poslužiti kao primjer za prikaz rada datotečnog podsustava. Konačno, ostvaren je i datotečni podsustav operacijskog sustava koji upravlja datotečnim sustavom *FSE* preko sučelja *fs_t*. Također, dano je i sučelje datotečnog podsustava prema primjenskim programima.

Pri izradi datotečnog podsustava pretpostavljeno je da može upravljati sa više različitih datotečnih sustava. Trenutno on omogućuje upravljanje datotečnim sustavom *FSE*, ali, ako je potrebno, mogao bi se proširiti.

5. Literatura

[1] Leo Budin, Marin Golub, Domagoj Jakobović, Leonardo Jelenković, *Operacijski sustavi*, 2.izdanje, Zagreb, Element, 2011.

[2] Leonardo Jelenković, *Operacijski sustavi za ugradbena računala*, 2013.

6. Sažetak

U ovom radu opisuje se ostvarenje pojednostavljenog datotečnog podsustava koji se može ugraditi u operacijski sustav za ugradbena računala. Ono ostvaruje osnovne operacije potrebne za rad sa datotekama. Kako bi se mogao prikazati njegov rad, ostvaren je i jednostavan datotečni sustav *FSE* te uređaj koji ga koristi. Uređaj je ostvaren u radnom spremniku.

Ključne riječi: datotečni podsustav, datotečni sustav, ugradbeno računalo

7. Abstract

This paper describes the realization of a simplified file subsystem that can be installed in operating systems for embedded computers. It implements the basic operations required for working with files. In order to show how it works, a simple file system *FSE* and the device that uses it are implemented. The device is implemented in RAM.

Keywords: file subsystem, file system, embeded computer