SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 379

Ostvarenje EDF raspoređivača za OSIER

Mario Iličić

Zagreb, lipanj 2012.

Umjesto ove stranice umetnite izvornik Vašeg rada.
Da bi ste uklonili ovu stranicu obrišite naredbu `\izvornik`.

SADRŽAJ

1. Uvod **1**

2. OSIER **2**

2.1. Struktura OSIER-a 2

2.2. Vrste raspoređivača kod OSIER-a 2

2.3. Ostvarenje dodatnih raspoređivača 3

3. EDF raspoređivanje **7**

3.1. Principi EDF raspoređivanja 7

3.2. OSIER i EDF 8

3.3. Komunikacija FIFO i EDF raspoređivača 9

3.3.1. Dolazak dretve u sustav 10

3.3.2. Promjena aktivne dretve 11

3.3.3. Alarm 11

3.3.4. Izlazak dretve iz sustava 11

4. Implementacija EDF raspoređivanja **12**

4.1. Sučelje prema programu - API 12

4.2. Prosljeđivanje poziva programa do EDF raspoređivača 13

4.3. EDF raspoređivač 13

4.3.1. Funkcija za inicijalizaciju raspoređivača 15

4.3.2. Funkcija za dodavanje dretve u sustav raspoređivača 15

4.3.3. Funkcija za uklanjanje dretve iz EDF sustava 15

4.3.4. Funkcija za aktiviranje dretve 16

4.3.5. Funkcija za deaktiviranje dretve 16

4.3.6. Funkcije za postavljanje i dohvaćanje parametara za raspoređivanje 16

4.3.7. Funkcija za postavljanje dretvinih parametara za raspoređivanje 16

4.3.8. Funkcija za dohvaćanje dretvinih parametara za raspoređivanje 17
4.3.9. Ostale funkcije potrebne za implementaciju raspoređivanja . . 17

5. Primjeri raspoređivanja **20**
5.1. Primjer 1 - Postavka EDF_TERMINATE 21
5.2. Primjer 2 - Postavka EDF_CONTINUE 23
5.3. Primjer 3 - Postavka EDF_SKIP 23

6. Zaključak **27**

Literatura **28**

A. Primjer 3 - Tablica testiranja **30**

1. Uvod

U ovom radu prikazano je ostvarenje raspoređivača prema krajnjim trenucima završetka zadataka (eng. *Earliest deadline first*, skraćeno EDF). Za njegovu implementaciju koristi se OSIER operacijski sustav. OSIER je operacijski sustav za edukaciju (eng. *Operating System Increments for Education and Research*, skraćeno OSIER), te se izgrađuje za akademske svrhe. Sustav nije potpuno izgrađen niti se tome teži. Glavni razlog tome, a ujedno i svrha samog sustava, je da se kroz njegovu izgradnju pojedine osobe, prvenstveno studenti, upoznaju sa pojedinim područjima koja su uobičajeni dijelovi jednog operacijskog sustava.

EDF raspoređivanje upravlja zadacima prema svojoj politici: uvijek odabire zadatak sa najbližim krajnjim trenutkom završetka, (Wikipedia, 2012). Na početku će biti nekoliko riječi o OSIER-u i raspoređivačima implementiranim u njemu. Zatim, da bi se moglo potpuno shvatiti na koji način zajedno funkcioniraju OSIER i EDF podsustav, bit će objašnjena komunikacija između ta dva sustava. Ne ulazi se u potpuno ostvarenje cijelog operacijskog sustava nego samo u ostvarenje EDF podsustava. Nakon toga slijedi detaljna analiza izgrađenog sustava raspoređivanja, i na kraju testiranje sustava kroz pojedine primjere koji su, osim riječima, predstavljeni slikama i tablicama.

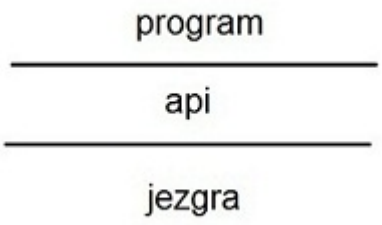
Kôd dosad izgrađenog sustava (sa uključenim EDF podsustavom) nalazi se na priloženom cd-u.

2. OSIER

Kako je već rečeno, a i kako se vidi iz imena, OSIER je operacijski sustav namijenjen za akademske svrhe. Nije u potpunosti izgrađen, međutim, ono što je dosad izgrađeno može se pokrenuti. Budući da je tema ovoga rada izgradnja EDF raspoređivača, izostavlja se detaljna analiza operacijskog sustava, osim s aspekta komunikacije samog operacijskog sustava i EDF podsustava.

2.1. Struktura OSIER-a

OSIER se, kao i svaki drugi operacijski sustav, sastoji od jezgre koja upravlja radom cijelog sustava te od sučelja (API) preko kojeg programi ulaze u interakciju sa jezgrom sustava (Slika 2.1).



Slika 2.1: Struktura OSIER-a

2.2. Vrste raspoređivača kod OSIER-a

Kao i svaki operacijski sustav, OSIER u svom radu ima za zadaću raspoređivanje različitih zadataka odnosno dretvi. Zadaci se mogu raspoređivati na razne načine. Kod OSIER-a postoje tri vrste raspoređivanja: raspoređivanje prema redu prispijeća (eng. *First In First Out*, skraćeno FIFO), posluživanje podjelom vremena (eng. *Round-*

Robin, skraćeno RR) i raspoređivanje prema trenucima krajnjeg završetka (eng. *Earliest Deadline First*, skraćeno EDF).

Raspoređivanje prema redu prispijeća je glavno raspoređivanje u cijelom sustavu. Sav posao koji FIFO obavlja je da iz globalnog reda pripravnih dretvi odabere dretvu najvećeg prioriteta, te nju proglašava aktivnom. Ukoliko u redu pripravnih postoji nekoliko dretvi istog prioriteta, FIFO ih odabire kako samo ime kaže - prema redu prispijeća.

Posluživanje podjelom vremena predstavlja dodatno raspoređivanje. RR raspoređivač dretvama dodijeli određeni kvant vremena, te one toliko obavljaju svoj posao. Kada kvant istekne, ako nisu obavile sav posao, stavljaju se na kraj reda istog prioriteta, dok se iduća dretva aktivira.

EDF raspoređivač također predstavlja dodatni raspoređivač. Svaka EDF dretva ima svoj TKZ - trenutak krajnjeg završetka (eng. *deadline*). TKZ je glavni parametar koji se uzima u obzir kod EDF posluživanja. Dretva sa najbližim TKZ-om se odabire i proglašava edf-aktivnom, (Lipari, 2005).

Kompletno raspoređivanje izvedeno je kombinacijom globalnog FIFO i dodatnih raspoređivača. Kada odabere dretvu sa najbližim TKZ-om, EDF je prebacuje u globalni red pripravnih, dok su RR dretve odmah sve u tom redu. Tamo dretve glavni raspoređivač poslužuje po redu prispijeća.

Prema tome, utjecaj dodatnih raspoređivača ostvaruje se promjenom prioriteta dretvi kojima ti raspoređivači upravljaju, dok se samo raspoređivanje dalje obavlja prema prioritetu (FIFO).

2.3. Ostvarenje dodatnih raspoređivača

Zajednička stvar svih dodatnih raspoređivača je korištenje istog sučelja. Usprkos tome, razlika je nekoliko. Prije svega, razlika im je sam princip raspoređivanja. Osim toga, razlikuju se u samom ostvarenju funkcija koje čine sučelje, te osim tih funkcija koriste i različite dodatne funkcije u ostvarenju raspoređivanja.

Sučelje svih dodatnih raspoređivača je isto, te je ono prikazano u nastavku:

```
typedef struct _ksched_t_ ksched_t;
struct _ksched_t_
{
        int sched_id;
        int (*init) ( ksched_t *self );
        int (*thread_add) ( kthread_t * );
```

```
        int (*thread_remove) ( kthread_t * );
        int (*thread_activate) ( kthread_t * );
        int (*thread_deactivate) ( kthread_t * );
        int (*set_sched_parameters) ( int sched_policy, sched_t *);
        int (*get_sched_parameters) ( int sched_policy, sched_t *);
        int (*set_thread_sched_parameters) ( kthread_t *, sched_t *);
        int (*get_thread_sched_parameters) ( kthread_t *, sched_t *);
        ksched_params_t params;
};
```

Osim osnovnih funkcija, sučelje sadrži još identifikator raspoređivača - `sched_id`, te strukturu `ksched_params_t params` koja sadrži globalne podatke potrebne raspoređivaču.

Budući da je tema rada implementacija EDF raspoređivanja, iduća poglavlja biti će posvećena tome, dok će u nastavku poglavlja kratko biti opisano samo ostvarenje drugog dodatnog raspoređivača.

Raspoređivanje podjelom vremena

Raspoređivanje podjelom vremena temelji se na principu da se svakoj dretvi dodijeli određeni kvant vremena. To vrijeme ona bude aktivna. Kada vrijeme istekne, stavlja se na kraj reda pripravnih dretvi istog prioriteta, te ponovno čeka svoj red. Tako se redom raspoređuju dretve istog prioriteta. Ako u sustavu u nekom redu istog prioriteta postoje dretve različitih politika raspoređivanja, raspoređuju se svaka na svoj način. Pretpostavimo da imamo u redu dretve istog prioriteta, a različitih politika raspoređivanja: 1.RR, 2.RR, 3. EDF, 4.RR. Prve dvije dretve će se raspoređivati prema RR politici, te će se njima dodijeliti po kvant vremena. Treća dretva će se izvršiti prema svojoj politici, dok će četvrta opet po RR principu. Naravno, RR dretve ukoliko ne završe svoje izvođenje u dodijeljenom kvantu, stavljaju se na kraj reda nakon isteka toga kvanta.

Iako imaju isto sučelje, RR raspoređivač ima različitu implementaciju funkcija iz tog sučelja od EDF raspoređivača. Dobar dio tih funkcija trenutno nije implementiran ili nema nikakvu svrhu.

To su sljedeće funkcije:

- funkcija za uklanjanje dretve iz RR sustava
  prototip: `int (*thread_remove) ( kthread_t * )`
- funkcija za dohvaćanje parametara za raspoređivanje

prototip: `int (*get_sched_parameters) ( int sched_policy, sched_t *)`

- funkcija za postavljanje dretvinih parametara raspoređivanja
  prototip: `int (*set_thread_sched_parameters) ( kthread_t *, sched_t *)`
- funkcija za dohvaćanje dretvinih parametara raspoređivanja
  prototip: `int (*get_thread_sched_parameters) ( kthread_t *, sched_t *)`

Najvažnije funkcije su funkcije za aktiviranje i deaktiviranje dretve. One obavljaju sav posao potreban za RR raspoređivanje. Funkcija za aktiviranje dretve poziva se kada je dretva odabrana za izvođenje, tj. kada postaje aktivna. Tada joj se dodijeli kvant vremena i postavlja se alarm. Kada se alarm oglasi, istekao je dodijeljeni kvant te se dretva stavlja na kraj reda za raspoređivanje. Funkcija za deaktiviranje provjerava da li je dretva "odradila" cijeli kvant vremena. Ako iz nekog razloga nije, provjerava se koliko joj je još ostalo, te se prema tome dretva stavlja na početak ili na kraj reda pripravnih. Gdje se stavlja ovisi o nekoj pretpostavljenoj vrijednosti (`threshold` - vidi dalje u tekstu) s kojom se "preostalo" vrijeme uspoređuje. Nakon što funkcija vrati povratnu vrijednost, dretva se deaktivira.

Prototipovi tih funkcija glase:

```
int (*thread_activate) ( kthread_t * );

int (*thread_deactivate) ( kthread_t * );
```

Osim te dvije funkcije, implementirane su još funkcije:

- funkcija za inicijalizaciju raspoređivača - inicijalizira raspoređivač i stvara alarm
- funkcija za dodavanje dretve u RR sustav - u opisnik dretve u strukturu podataka za raspoređivanje postavlja vremenski kvant
- funkcija za postavljanje parametara raspoređivanja - u vlastitu strukturu raspoređivača postavlja vremenski kvant

Prototipovi tih funkcija su redom:

```
int (*init) ( ksched_t *self );
int (*thread_add) ( kthread_t * );
int (*set_sched_parameters) ( int sched_policy, sched_t *);
```

Osim tih osnovnih funkcija koje čine sučelje, postoji još jedna pomoćna funkcija koja predstavlja reakciju na alarm. Budući da se ona poziva na istek kvanta, briše vrijeme dodijeljeno dretvi i prebacuje ju na kraj reda. Na kraju poziva globalnu funkciju za raspoređivanje dretvi (FIFO) - `kthreads_schedule()`.

Prototip te funkcije glasi:

```
static void rr_timer ( void *p )
```

Osim funkcija, sučelje raspoređivača sadrži strukturu podataka koja sadrži parametre raspoređivača, potrebne za raspoređivanje. Ta struktura prikazana je u nastavku:

```
typedef struct _ksched_rr_t_
{
        time_t time_slice;
        time_t threshold;

        void *rr_alarm;
        alarm_t alarm;
}
ksched_rr_t;
```

gdje `time_slice` predstavlja vrijeme koje se dodjeljuje dretvama, `threshold` predstavlja vrijednost s kojom se uspoređuje preostalo vrijeme dretve, a `*rr_alarm` je pokazivač na alarm koji označava istek kvanta.

3. EDF raspoređivanje

EDF raspoređivanje pripada algoritmima s dinamičkim dodjeljivanjem prioriteta, (Lipari, 2005). To znači da se prioritet zadataka može mijenjati tijekom izvršavanja. Prioritet kod EDF dretvi je obrnuto propocionalan dretvinom TKZ-u, tj što je TKZ bliži, prioritet je veći.

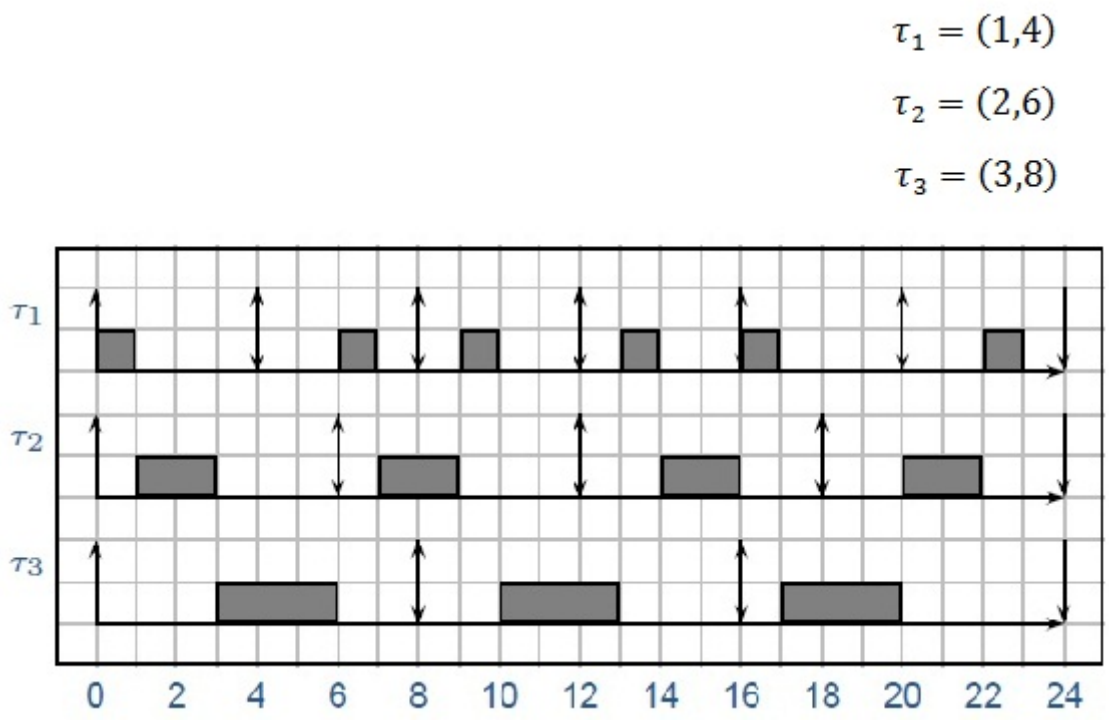
3.1. Principi EDF raspoređivanja

EDF posluživanje ostvaruje se uvijek prema istim principima. Svaka dretva ima nekoliko parametara (opisnik), međutim, samo su dva koja se odnose isključivo na EDF raspoređivanje. To su TKZ i period. Period predstavlja trenutke kada je periodična dretva ponovno spremna za obavljanje svoga posla, dok TKZ predstavlja trenutak u periodu do kad dretva mora završiti svoj posao. U radu raspoređivač uvijek bira dretvu sa najbližim TKZ-om, te ona postaje aktivna. Slika 3.1 prikazuje primjer jednog EDF raspoređivanja (u zagradi prva vrijednost predstavlja TKZ, a druga period).

Na samom početku, sve tri dretve su pripravne, te se odabire D1 jer ima najbliži TKZ. Kada ona odradi svoj posao u prvom periodu, odabire se D2 jer od preostalih dretvi ona ima najbliži TKZ. Nakon D2 slijedi D3. Raspoređivanje se nastavlja dalje na isti način.

Prednosti EDF-a

1. nije potrebno definirati prioritete
2. manje zamjena konteksta - potrebno je manje zamjena konteksta nego kod drugih raspoređivača
3. velika iskorištenost procesora



Slika 3.1: Jednostavan primjer EDF raspoređivanja

Nedostaci EDF-a

1. manje upravljiv - manja kontrola nad izvršavanjem
2. kompleksnija implementacija - zahtijeva provedbu detaljnih mjerenja svih vremenskih parametara dretvi, (Jelenković, 2012).

3.2. OSIER i EDF

EDF raspoređivač za sustav OSIER predstavlja dodatni raspoređivač, zajedno sa RR raspoređivanjem. Kako je već rečeno, dretve se raspoređuju prema krajnjem završetku. Svaka dretva ima svoje EDF parametre prema kojima se raspoređuje. Kod ovoga ostvarenja ti parametri se na samom početku pohrane u opisnik dretve. U trenucima kada se bira dretva koja će se aktivirati, bira se ona sa najbližim krajnjim završetkom, (Jelenković, 2011). Kada se ona izabere, proglašava se EDF-aktivnom, te se stavlja u globalni red pripravnih. Tamo ostaje dok se ne dogodi jedna od sljedeće dvije situacije:

– dretva završava svoje periodičko izvođenje,

– pojavila se druga dretva, koja ima bliži TKZ, te je izabrana i proglašena aktivnom

U prvom slučaju dretva odlazi u red čekanja u EDF sustavu, dok u drugom se vraća u EDF red pripravnih. Dretva u redu čekanja ostaje sve do trenutka njenog ponovnog aktiviranja. Kada dođe taj trenutak, aktivira se alarm za period, te se dretva prebacuje u red pripravnih. Svaki put kada neka dretva dođe u red pripravnih, ponovno se provjerava koja ima najbliži TKZ, te se ona izabire kao aktivna.

Osim tih osnovnih principa, vrlo je važno što se događa u slučaju kad dretva ne uspije završiti prije TKZ-a. U takvoj situaciji poduzimaju se akcije ovisno o kontrolnoj zastavici dretve. Postoje 3 različite vrijednosti za kontrolne zastavice o kojima ovisi izvođenje dretve:

– `EDF_TERMINATE`
– `EDF_CONTINUE`
– `EDF_SKIP`

Ako se tijekom izvršavanja dretve dogodi da nije završila prije TKZ-a, a postavljena je kontrolna zastavica `EDF_TERMINATE`, uklanjaju se svi alarmi, te se dretva potpuno "ubija". Ako je, pak, postavljena kontrolna zastavica `EDF_CONTINUE`, dretva će nastaviti svoj rad te tako prekoračiti preko TKZ-a. Samim prekoračenjem može dalje uzrokovati prekoračenje TKZ-a drugih dretvi. Ako je dretva prekoračila TKZ dok je postavljena kontrolna zastavica `EDF_SKIP`, "preostali posao" se prebacuje na idući period. To prebacivanje uzrokuje i ažuriranje svih potrebnih struktura i varijabli (alarmi, period, TKZ, `next_run`, `active_deadline`, itd.).

3.3. Komunikacija FIFO i EDF raspoređivača

Da bi se dretve mogle raspoređivati kako je prethodno opisano, mora postojati komunikacija između EDF podsustava i glavnog sustava raspoređivanja (Slika 3.2).

Komunikacija se ostvaruje pomoću poziva jezgrinih funkcija. Sami pozivi tih funkcija događaju se u karakterističnim trenucima. Postoje četiri takva trenutka:

1. dolazak dretve u sustav
2. promjena aktivne dretve
3. alarm koji je postavio EDF

Slika 3.2: Ostvarenje dodatnih raspoređivača

4. izlazak dretve iz sustava

Sama EDF dretva izgleda kako je prikazano u nastavku:

```
dretva( parametri )
{
        edf_set( parametri );

        while( uvjet )
        {
                edf_wait();

                radi_nešto;
        }
        edf_exit;
}
```

Nakon što se stvori, dretva poziva funkciju `edf_set`. Tim pozivom postavljaju se parametri u EDF sustav, te se stvaraju potrebni alarmi za tu dretvu. Nakon toga dretva u `while` petlji obavlja svoj posao. Repetitivno poziva funkciju `edf_wait` preko koje komunicira sa raspoređivačem. Na prvi poziv te funkcije dretva se postavlja u EDF red pripravnih dretvi. Kada završi cjelokupni posao, tj. kada izađe iz petlje, dretva poziva funkciju `edf_exit` koja uklanja alarme, uklanja dretvu iz EDF sustava (mijenja joj politiku raspoređivanja), te poziva glavni raspoređivač FIFO.

3.3.1. Dolazak dretve u sustav

Kako je prethodno rečeno, kada se stvori dretva čija je politika raspoređivanja EDF, ona preko API-ja poziva jezgrinu funkciju koja dalje preko EDF sučelja komunicira

sa samim raspoređivačem, te dretva biva dodana u EDF sustav. Točnije, postavljaju se parametri i stvaraju i postavljaju se alarmi.

3.3.2. Promjena aktivne dretve

Nakon što dretva dođe u red pripravih u EDF sustavu, pregledava se da li trenutna aktivna ima i dalje najbliži TKZ. Ako nema, bira se nova dretva (sa najbližim TKZ-om) te se proglašava aktivnom. Kada se proglasi aktivnom, dretvu se prebacuje u globalni red pripravnih, tj. komunicira se sa glavnim sustavom. Kada joj istekne predviđeno vrijeme, vraća se u EDF sustav u red čekanja. Tamo se nalazi dok ne bude probuđena.

3.3.3. Alarm

Svaka dretva ima dva alarma, jedan za period, drugi za TKZ. Ako se dretva nalazi u redu čekanja, na alarm za period prelazi u red pripravnih. Čim pređe, ponavlja se scenarij iz prethodne situacije. Ako dretva nije periodička, ni alarm se neće aktivirati. Drugi alarm se događa ako dretva ne završi prije trenutka krajnjeg završetka, te se ovisno o kontrolnim zastavicama poduzimaju određene akcije.

3.3.4. Izlazak dretve iz sustava

Četvrti karakterističan trenutak kada dolazi do komunikacije EDF podsustava i globalnog sustava je kada dretva izlazi iz sustava. U tom slučaju, kao i pri dolasku u sustav, preko sučelja poziva jezgru. Jezgra tada poziva raspoređivač, koji dretvu uklanja iz svog sustava te poziva glavni raspoređivač.

Nakon teorijskog uvoda, slijedi opis programske implementacije.

4. Implementacija EDF raspoređivanja

Implementacija EDF raspoređivanja može se podijeliti na tri dijela:

- sučelje prema programu (API)
- proslijeđivanje poziva programa do EDF raspoređivača
- EDF raspoređivač

4.1. Sučelje prema programu - API

Kada neki program želi komunicirati sa raspoređivačem, poziva funkcije sučelja prema programu - funkcije API-ja. Postoji nekoliko različitih funkcija. Ipak, onih koji su potrebni za komunikaciju sa EDF raspoređivačem ima samo 4. Prva funkcija je funkcija za stvaranje dretve. Program ju poziva, te joj prosljeđuje parametre dretve. Njen protopip glasi:

```
int create_thread ( void *start_func, void *param,
                    int sched, int prio,
                    thread_t *handle );
```

Nakon što se dretva stvori, pomoću tri preostale funkcije poziva raspoređivač, te se tako ostvaruje njeno raspoređivanje.

```
int edf_set ( time_t deadline, time_t period, int flags );
int edf_wait ();
int edf_exit ();
```

Kada se dretva stvori, poziva prvu funkciju - `edf_set`, kada se krene izvršavati poziva funkciju `edf_wait`, a kada izlazi iz sustava funkciju `edf_exit`. Kada se pozovu te tri funkcije, one postavljaju svoje zastavice koje određuju daljnji tijek događaja

(EDF_SET, EDF_WAIT, EDF_EXIT). Taj tijek opisan je kod funkcije za postavljanje dretvinih parametara za raspoređivanje, potpoglavlje 4.3.7.

4.2. Prosljeđivanje poziva programa do EDF raspoređivača

Kada se pozove neka od prethodno navedenih funkcija API-ja, raspoređivač se ne poziva direktno, nego je to ostvareno preko poziva sustavskih funkcija. Kod EDF-a su iskorištene dvije takve funkcije. Prva je upotrebljena kod stvaranje dretve, dok druga ostvaruje pozive raspoređivača od sve tri funkcije za komunikaciju sa EDF-om. Prototipovi tih funkcija glase:

```
int sys__create_thread ( void *p );
int sys__set_thread_sched_params ( void *p );
```

Prva funkcija prima paametre preko stoga, učitava parametre u svoje varijable te stvara dretvu. Druga funkcija, također prima parametre preko stoga, učitava ih u svoje varijable, postavlja politiku raspoređivanja za dretvu te preko sučelja raspoređivača poziva funkciju preko koje se ostvaruje komunikacija sa raspoređivačem - `set_thread_sched_parameters`.

4.3. EDF raspoređivač

Svaka dretva, kojoj je politika raspoređivanja EDF, preko sučelja (API) poziva jezgru, koja poziv prosljeđuje raspoređivaču. EDF raspoređivač ima svoje sučelje preko kojeg komunicira sa ostatkom sustava. Osim sučelja, raspoređivač sadrži svoju globalnu strukturu podataka pomoću koje ostvaruje raspoređivanje. Ta struktura podataka sadrži red pripravnih, red čekanja i zastavicu na trenutno aktivnu dretvu:

```
typedef struct _ksched_edf_t_
{
        kthread_t *active;
        kthread_q ready;
        kthread_q wait;
}
ksched_edf_t;
```

Kako je već rečeno kod RR, sučelje dodatnih raspoređivača je isto, te ono sadrži pokazivače na sve funkcije/akcije koje se mogu izazvati izvan raspoređivača, već prikazano na stranici 3.

Osim tih kazaljki, u opisniku se nalazi identifikator raspoređivača `sched_id`, kao i struktura podataka `ksched_params_t` koja sadrži prethodno navedenu strukturu podataka potrebnu za sam raspoređivač.

Svaka dretva koja se nalazi u EDF sustavu ima u svom opisniku strukturu podataka koja sadrži parametre potrebne za raspoređivanje. Struktura sa parametrima je prikazana u nastavku:

```
typedef struct _ksched_edf_thread_params_t
{
        time_t relative_deadline;
        time_t period;
        time_t next_run;
        time_t active_deadline;
        int flags;

        void *edf_period_alarm;
        void *edf_deadline_alarm;
}
ksched_edf_thread_params_t;
```

gdje su:

- `period` - trenutak kad dretva bude ponovno aktivirana (ako je periodična)
- `relative_deadline` - vrijednost koja predstavlja TKZ
- `active_deadline` - trenutak dokad dretva mora završiti u tekućem periodu
- `next_run` - trenutak sljedećeg aktiviranja dretve (ažurira se periodom)
- `flags` - zastavice za upravljanje raspoređivanjem
- `*edf_period_alarm` - pokazivač na alarm za period
- `*edf_deadline_alarm` - pokazivač na alarm za TKZ

U nastavku slijedi opis funkcija raspoređivača.

4.3.1. Funkcija za inicijalizaciju raspoređivača

Funkcija `init` poziva se za inicijalizaciju globalnih podataka svakog raspoređivača. Već je rečeno da su globalni podaci za ostvarenje EDF raspoređivanja dva reda u kojima se dretve nalaze te pokazivač na aktivnu dretvu (struktura prikazana na 13. stranici).

Prototip funkcije:

```
static int edf_init ( ksched_t *self );
```

4.3.2. Funkcija za dodavanje dretve u sustav raspoređivača

Ova funkcija poziva se kada je dretva stvorena, ili kada je dretvi za politiku raspoređivanja odabrana EDF politika. Samo dodavanje dretve u EDF sustav (u red `EDF.ready`) ostvareno je u drugoj funkciji, tako da ova funkcija samo dohvaća alarm dretve i inicijalizira ga na NULL vrijednost.

Prototip funkcije:

```
static int edf_thread_add ( kthread_t *kthread );
```

`kthread` je kazaljka na dretvu koja se dodaje.

4.3.3. Funkcija za uklanjanje dretve iz EDF sustava

Pozivom funkcije `thread_remove`, dretva se uklanja iz EDF sustava. Prilikom uklanjanja dretve, obavljaju se sljedeće akcije:

- ako kazaljka na aktivnu dretvu u EDF sustavu pokazuje na dretvu koja se uklanja, kazaljka se briše (postavlja na NULL)
- uklanjaju se oba alarma, te
- mijenja se politika raspoređivanja

Prototip funkcije:

```
static int edf_thread_remove ( kthread_t *kthread );
```

4.3.4. Funkcija za aktiviranje dretve

Ova funkcija nije implementirana kod EDF raspoređivača. Inače je predviđena za aktiviranje dretve, međutim, u EDF sustavu to je ostvareno kroz druge mehanizme.

Prototip funkcije:

```
static int edf_thread_activate ( kthread_t *kthread );
```

4.3.5. Funkcija za deaktiviranje dretve

Slično kao i prethodna, `thread_deactivate` funkcija praktički nije implementirana. Sav "posao" koji ona obavlja je da pozove dodatnu funkciju `k_edf_schedule()`, koja, nakon deaktiviranja dretve, bira novu dretvu za aktiviranje (sa najbližim TKZ-om).

Prototip funkcije:

```
static int edf_thread_deactivate ( kthread_t *kthread );
```

4.3.6. Funkcije za postavljanje i dohvaćanje parametara za raspoređivanje

Ove funkcije predviđene su za postavljanje i dohvaćanje parametara raspoređivača. Te akcije kod EDF-a se ostvaruju na drugi način, tako da ni ove funkcije nisu implementirane.

Prototipovi funkcija:

```
static int edf_set_sched_parameters ( int sched_policy, sched_t *params );

static int edf_get_sched_parameters ( int sched_policy, sched_t *params );
```

4.3.7. Funkcija za postavljanje dretvinih parametara za raspoređivanje

Funkcija `set_thread_sched_parameters` jedna je od važnijih funkcija u ostvarenju EDF raspoređivanja. Preko nje dretve "komuniciraju" sa raspoređivačem. Do te komunikacije dolazi u tri različite situacije, tako da ona ima tri različita dijela. Zastavica postavljena u varijabli `params->edf.flags` određuje koji dio funkcije će

se izvoditi. Zastavice se postavljaju kod pozivanja API-ja, te su moguće tri različite zastavice : EDF_SET, EDF_WAIT i EDF_EXIT.

Prva situacija (EDF_SET) je na samom početku kada se dretva stvori, te preko API-ja poziva raspoređivač. Tim pozivom postavlja parametre u opisnik dretve, stvara alarme za TKZ i period, te postavlja alarm za period. Budući da se u drugom dijelu vrijednosti uvijek ažuriraju na idući period, parametri alarma za period se postavljaju na "nulti" period. Cilj takvog postavljanja je da se u prvom prolazu kroz drugi dio funkcije vrijednosti postave na prvi period tako da dretva odmah bude spremna u tom prvom periodu.

Druga situacija (EDF_WAIT) je kada dretva krene u izvođenje, te opet preko API-ja poziva ovu funkciju. Ovaj dio funkcije ažurira parametre za periodni alarm te postavlja vrijednosti za alarm od TKZ-a. Osim toga, postavlja dretvu u jedan od EDF redova - `EDF.ready` ili `EDF.wait`, ovisno o tome da li je došao trenutak kad dretva postaje spremna.

Treća situacija (EDF_EXIT) se događa kad dretva izlazi iz EDF sustava. U tom slučaju, uklanjaju se alarmi te se dretvi mijenja politika raspoređivanja.

Prototip funkcije:

```
static int edf_set_thread_sched_parameters (kthread_t *kthread,
                                            sched_t *params);
```

4.3.8. Funkcija za dohvaćanje dretvinih parametara za raspoređivanje

Kao i nekoliko prethodno navedenih funkcija, i ova funkcija nije implementirana.

Prototip funkcije:

```
static int edf_get_thread_sched_parameters (kthread_t *kthread,
                                            sched_t *params);
```

4.3.9. Ostale funkcije potrebne za implementaciju raspoređivanja

Prethodno opisane funkcije čine sučelje raspoređivača te predstavljaju osnovne funkcije. Osim njih postoje i dodatne funkcije koje nisu manje važne od osnovnih. Te funkcije slijede u nastavku.

Funkcija za stvaranje i postavljanje alarma za period dretve

Funkcija služi za stvaranje i postavljanje alarma za period dretve. Poziva se u prvom dijelu funkcije `set_thread_sched_parameters` kada se želi stvoriti i postaviti alarm za period.

Prototip funkcije:

```
static int edf_arm_period ( kthread_t *kthread );
```

Funkcija za stvaranje alarma za TKZ dretve

`edf_deadline_period` stvara alarm za TKZ. Poziva se isto kad i prethodna funkcija.

Prototip funkcije:

```
static int edf_arm_deadline ( kthread_t *kthread );
```

Funkcija za odabir dretve sa najbližim TKZ-om

Funkcija predstavlja jednu od najvažnijih funkcija - ostvaruje osnovni princip EDF raspoređivanja: pregledava dretve te pronalazi onu sa najbližim TKZ-om, postavlja ju kao EDF-aktivnu, te ju stavlja u globalni red pripravnih.

Prototip funkcije:

```
static int k_edf_schedule ();
```

Funkcija - akcija na periodni alarm

Funkcija predstavlja akciju na istek alarma za period. Kada se "oglasi" alarm za period, počinje novi period za dretvu, te ona ponovno postaje spremna za izvođenje. Dretva se tada prebacuje iz reda `EDF.wait` u red `EDF.ready`. Jedina iznimka je ako se "oglasi" alarm za novi period, a dretva nije gotova u prethodnom. U tom slučaju se ništa ne događa, nego akcije poduzima alarm za TKZ budući da TKZ ne smije biti veći od perioda (mora biti gotov prije perioda).

Prototip funkcije:

```
static void edf_period_timer ( void *p );
```

Funkcija - akcija na alarm za TKZ

Ova funkcija predstavlja akciju na istek alarma za TKZ. On se oglašava u trenutku u kojem dretva mora biti gotova u tekućem periodu. Naravno, ako dretva završi prije toga trenutka, alarm se neće oglasiti, nego će se ažurirati za idući period dretve i istek TKZ-a u tom periodu. Ako se ipak alarm oglasi, dretva nije uspjela završiti dokad je trebala, te nastavak njenog izvođenja ovisi o kontrolnim zastavicama postavljenim u opisniku dretve. Te zastavice i ono što se događa ovisno o njima, opisano je u trećem poglavlju (3.2. OSIER i EDF).

Prototip funkcije:

```
static void edf_deadline_timer ( void *p );
```

Funkcija za provjeru prekoračenja TKZ-a

Funkcija služi za provjeru da li je dretva prekoračila TKZ, tj. da li je prošao trenutak do kojeg je trebala završiti izvođenje.

Prototip funkcije:

```
static int edf_check_deadline ( kthread_t *kthread );
```

5. Primjeri raspoređivanja

Za potrebe ispitivanja napravljen je program koji treba prilagoditi odredišnom sustavu obzirom da se "posao" simulira. Program stvara EDF dretve koje dalje komuniciraju sa EDF raspoređivačem na način opisan u prethodnim poglavljima. Sažeti pseudokôd EDF dretvi prikazan je na 10. stranici.

Prilikom testiranja dvije stvari uzete su u obzir. Prije svega, to su kontrolne zastavice koje odlučuju o bitnim stvarima tijekom raspoređivanja. Već je rečeno da postoje tri takve zastavice čije su funkcije poznate: EDF_TERMINATE, EDF_CONTINUE i EDF_SKIP. Druga bitna stvar je omjer veličina perioda i TKZ-a. Logično je da je period veći od ili jednak TKZ-u. U programu koji testira EDF sustav implementirana su dva omjera:

1. TKZ = period
2. TKZ = (1/2) * period

Testiranje je provedeno na nekoliko primjera. Ovdje će biti prikazana tri primjera - za svaku kontrolnu zastavicu po jedan, uz uvjet da je period jednak TKZ-u. Primjeri će pokazati da je sustav uspješno implementiran. Ostali primjeri, sa svim ispisima programa, se nalaze na priloženom cd-u.

Kod svakog primjera stvaraju se četiri dretve, kojima se dodjeljuje određeno vrijeme kada postanu aktivne (u ovim primjerima je to otprilike oko 0.5 i 0.6 sekundi). Ovisno o tome vremenu dretve će se (ne)uspješno raspoređivati. Svako testiranje traje 20 sekundi.

Radi jednostavnosti period je određen rednim brojem dretve. Prema tome dobijemo sljedeće vrijednosti (u sekundama):

- za omjer TKZ = period:
    1. Dretva 1: period = 1, TKZ = 1

2. Dretva 2: period = 2, TKZ = 2
3. Dretva 3: period = 3, TKZ = 3
4. Dretva 4: period = 4, TKZ = 4

– za omjer TKZ = (1/2) * period:

1. Dretva 1: period = 1, TKZ = 0.5
2. Dretva 2: period = 2, TKZ = 1
3. Dretva 3: period = 3, TKZ = 1.5
4. Dretva 4: period = 4, TKZ = 2

5.1. Primjer 1 - Postavka EDF_TERMINATE

U trećem poglavlju opisana je uloga zastavice EDF_TERMINATE: kada dretva ne završi do trenutka kada bi trebala (TKZ), ona se "ubija". Raspoređivanje primjera sa kontrolnom zastavicom EDF_TERMINATE prikazano je na slici 5.1. Na slici gornje strelice pokazuju trenutke kada dretve postaju spremne i dolaze u red EDF pripravnih, dok donje strelice pokazuju trenutke kada završavaju sa izvođenjem te prelaze u red čekanja. Jedina su iznimka strelice crvene boje - one označavaju da je došlo do prekoračenja TKZ-a, samim time i do "ubijanja" dretvi.

Na samom početku sve 4 dretve su spremne, nalaze se u `EDF.ready`, te se izabire dretva 1 (D1) budući da ima najbliži TKZ. U trenutku t = 0.5 ona završava, odlazi u `EDF.wait`, a za aktivnu se bira dretva 2 (D2) jer sada ona ima najbliži TKZ. Dok D2 radi, u t = 1 aktivira se alarm za period D1, te ona prelazi u `EDF.ready`. Kako je došla u red pripravnih, provjerava se koja dretva ima najbliži TKZ. Najbliži TKZ ima D1 pa ona ponovno postaje aktivna, dok se D2 prekida. U t = 1.5 D1 završava, te se bira nova dretva sa najbližim TKZ-om. To je sada D2, pa ona nastavlja sa radom. Ona završava u t = 1.6, a nakon nje se odabire dretva D3. U t = 2 "bude" se D1 i D2, te D1 smjenjuje D3. Kada D1 završi, u t = 2.5, D3 nastavlja sa radom i završava u t = 2.6. Nakon nje ponovno je odabrana D2. U t = 3 oglašavaju se alarmi od D1 i D3, koje prelaze u `EDF.ready`. Zbog bližeg TKZ-a D1 smjenjuje D2. Završetkom D1, D2 nastavlja sa radom sve do t = 3.7 kada završava. Tada prvi puta aktivna postaje D4. Međutim, u t = 4 aktiviraju se njeni alarmi za period i TKZ. Budući da nije završila do t

Slika 5.1: Primjer sa EDF_TERMINATE zastavicom

= 4 i budući da je postavljena zastavica EDF_TERMINATE, D4 se prekida i "izbacuje" iz sustava. Osim tih alarma, aktiviraju se i alarmi za period od D1 i D2, te D1 postaje aktivna.

Na taj način raspoređivanje se nastavlja i dalje sve do t = 20 kada je prekid testiranja. Od toga preostalog raspoređivanja valja izdvojiti situaciju u t = 5.6. Za aktivnu je odabrana D3 koja je postala spremna još u t = 3, tako da joj je istek TKZ-a u t = 6. Budući da ne uspijeva završiti do toga trenutka, i ona se terminira, pa u sustavu ostaju samo dretve D1 i D2.

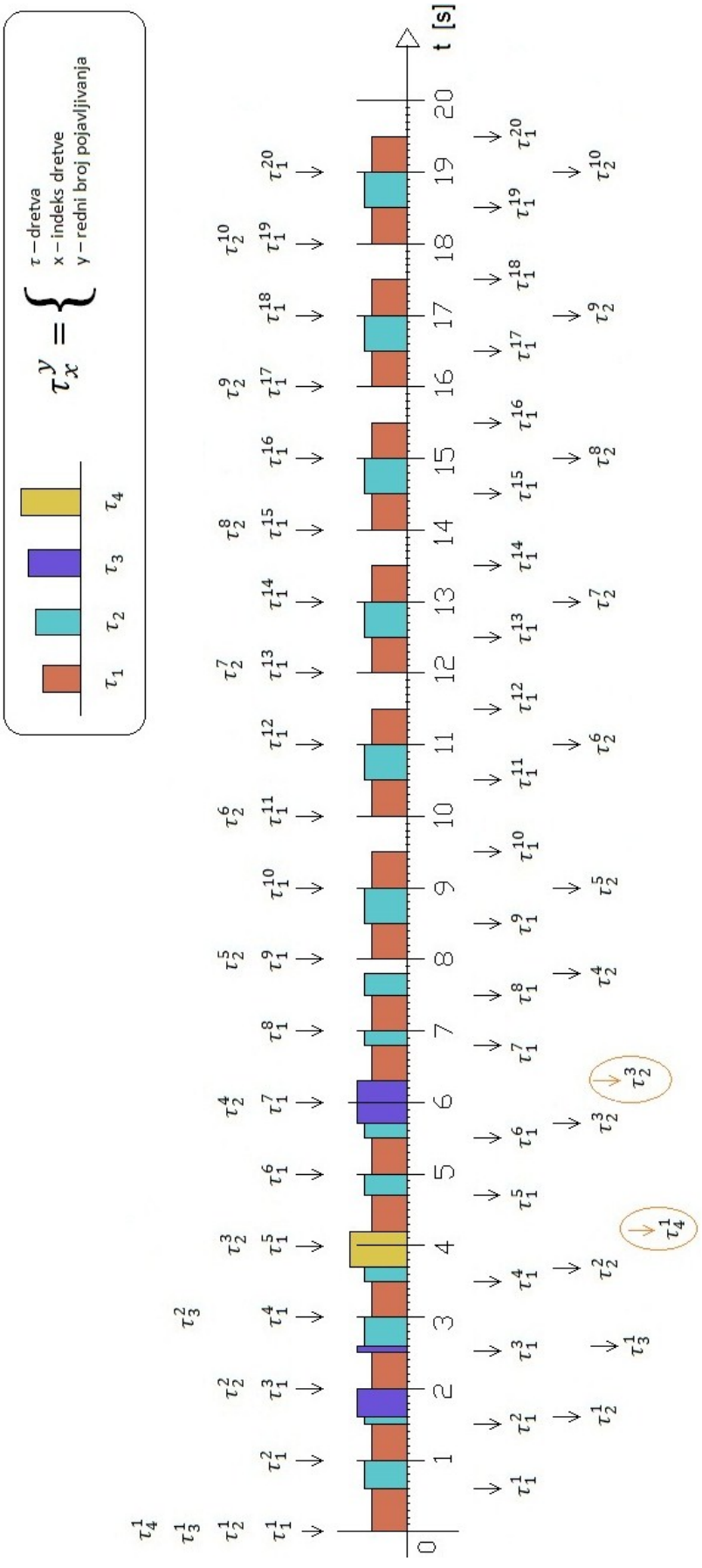
5.2. Primjer 2 - Postavka EDF_CONTINUE

Drugi primjer jako je sličan prvom. Dretve se raspoređuju prema istom principu. Jedina razlika je u kontrolnoj zastavici. Slika 5.2 prikazuje raspoređivanje sa zastavicom EDF_CONTINUE. Dva zanimljiva trenutka događaju se, jednako kao i u prvom primjeru, u t = 3.7 i t = 5.7. U prvom, D4 prvi put je izabrana za EDF aktivnu. Međutim, TKZ joj je do t = 4, tako da ne može stići obaviti posao prije isteka. Ipak, zbog kontrolne zastavice dopušteno joj je nastaviti dok ne završi posao, te nakon toga izlazi iz sustava, kako i pokazuje crvena strelica kod trenutka t = 4.2 gdje D4 završava sa radom. I u drugom zanimljivom trenutku D3 ne stiže obaviti posao do isteka TKZ-a, ali ipak zastavica joj dopušta da završi svoj posao. Nakon toga i ona kao i D4 napušta sustav, tj izlazi iz EDF sustava.

5.3. Primjer 3 - Postavka EDF_SKIP

Treći primjer odnosi se na kontrolnu zastavicu EDF_SKIP. Njena funkcija je također ranije navedena: ako dretva nije uspjela završiti do isteka TKZ-a, izvođenje joj se prebacuje u idući period. Raspoređivanje ovog primjera prikazano je tablicom 5.1. Sama tablica ima 5 stupaca:

- **t** - vrijeme (vremenski trenuci)
- **opis** - opis događaja u pojedinom trenutku
- **aktivna** - stupac koji pokazuje koja je dretve aktivna u određenom trenutku
- **edf.ready** - red EDF pripravnih dretvi
- **edf.wait** - red čekanja EDF dretvi



Slika 5.2: Primjer sa EDF_CONTINUE zastavicom

Tablica prikazuje događaje u pojedinim vremenski trenucima. Tih događaja ima nekoliko:

- DOLAZAK - dolazak dretve u EDF sustav
- KRAJ - događaj završetka izvođenja dretve u tekućem periodu
- P_ALARM - događaj alarma za period od dretve
- D_ALARM - događaj alarma za TKZ

Prema opisima događaja može se već naslutiti kako "čitati" tablicu. Nakon što bude stvorena, dretva DOLAZI u EDF sustav. Kada je dretva spremna i odabrana da bude aktivna, ona kreće s radom. Kada obavi svoj posao, došao je njen KRAJ. Novi periodi kada dretva ponovno postaje aktivna označeni su sa P_ALARM; tada ona prelazi iz reda čekanja u red pripravnih. Kada u red pripravnih dođe dretva sa bližim TKZ-om od trenutno aktivne, smjenjuje ju. Ako dretva ne uspije završiti prije isteka svog TKZ-a, pokušava prekoračiti taj trenutak. Taj istek označava D_ALARM. Budući da se radi o kontrolnoj zastavici EDF_SKIP, umjesto prekoračenja, njen posao se prebacuje u idući period. Svaka dretva koja je bila prekinuta, kada ponovno dođe njen red, nastavlja sa radom.

Budući da je tablica jako velika, ovdje je prikazan samo jedan njen dio. Cijela tablica prikazana je u dodatku (Tablica A.1).

Tablica 5.1: Raspoređivanje kada je postavljena zastavica EDF_SKIP

t [s]	opis	aktivna	edf.ready	edf.wait
0	DOLAZAK D1	–	D1	–
0	DOLAZAK D2	–	D1 D2	–
0	DOLAZAK D3	–	D1 D2 D3	–
0	DOLAZAK D4	–	D1 D2 D3 D4	–
0,6	KRAJ D1	–	D2 D3 D4	D1
1	P_ALARM D1	D2	D3 D4 D1	–
1,5	KRAJ D1	–	D3 D4 D2	D1
1,6	KRAJ D2	–	D3 D4	D1 D2
2	P_ALARM D1	D3	D4 D1	D2
2	P_ALARM D2	D1	D4 D3 D2	–

Nastavlja se na idućoj strani...

Tablica 5.1 – Nastavak

t [s]	opis	aktivna	edf.ready	edf.wait
2,5	KRAJ D1	–	D4 D3 D2	D1
2,6	KRAJ D3	–	D4 D2	D1 D3
3	P_ALARM D1	D2	D4 D1	D3
3	P_ALARM D3	D1	D4 D2 D3	–
3,5	KRAJ D1	–	D4 D2 D3	D1
3,7	KRAJ D2	–	D4 D3	D1 D2
4	P_ALARM D1	D4	D3 D1	D2
4	P_ALARM D2	D4	D3 D1 D2	–
4	P_ALARM D4	D4	D3 D1 D2	–
4	D_ALARM D4	D4	D3 D1 D2	–
4,6	KRAJ D1	–	D3 D2 D4	D1

6. Zaključak

U ovom radu prikazano je ostvarenje raspoređivača prema krajnjim trenucima završetaka zadataka - EDF. Sustav ima 3 raspoređivača: globalni FIFO i dodatne EDF i RR. Svi principi EDF raspoređivanja su uzeti u obzir prilikom implementacije. Sama implementacija ima sličnosti sa već postojećim RR raspoređivačem. Ta sličnost odnosi se na sučelje raspoređivača. Takva sličnost i mora postojati jer je predviđeno da svi dodatni raspoređivači imaju isto sučelje. Ipak, puno su veće razlike. Osim osnovnih funkcija iz sučelja, za ispravno raspoređivanje dodano je nekoliko funkcija, te je na taj način raspoređivanje upotpunjeno.

Uspješna implementacija potvrđena je kroz tri primjera - tri različite reakcije na karakterističan događaj prilikom raspoređivanja. Radi se o situaciji kada dretva ne uspije završiti prije isteka TKZ-a, te daljnji tijek raspoređivanja ovisi o kontrolnim zastavicama. Prema tome, i same kontrolne zastavice imaju velik utjecaj na raspoređivanje. Reakcije na sve tri zastavice su onakve kakve su i zamišljene, što je potvrda uspješne implementacije.

LITERATURA

Leonardo Jelenković. *Sustavi za rad u stvarnom vremenu*. Skripta za predavanja. Fakultet elektrotehnike i računarstva, Zavod za elektroniku, mikroelektroniku, računalne i inteligentne sustave, Sveučilište u Zagrebu, 2011. URL `http://www.fer.unizg.hr/_download/repository/T01_Uvod_u_SRSV.pdf`.

Leonardo Jelenković. *Operacijski sustavi za ugrađena računala*. Skripta za predavanja (u izradi). Fakultet elektrotehnike i računarstva, Zavod za elektroniku, mikroelektroniku, računalne i inteligentne sustave, Sveučilište u Zagrebu, 2012. URL `http://www.fer.unizg.hr/_download/repository/OSZUR_2012.pdf`.

Giuseppe Lipari. *Earliest Deadline First*. Scuola Superiore Sant'Anna, Pisa - Italy, 2005. URL `http://retis.sssup.it/~lipari/courses/str06/10.edf.pdf`.

Wikipedia. *Earliest deadline first scheduling*, 2012. URL `http://en.wikipedia.org/wiki/Earliest_deadline_first_scheduling`.

Ostvarenje EDF raspoređivača za OSIER

Sažetak

OSIER je operacijski sustav namijenjen za obrazovne svrhe, da se na njemu prikažu principi izgradnje komponenti operacijskog sustava za ugrađena računala. Zadatak rada je bio implementirati EDF raspoređivač, te ga testirati na nekoliko primjera. Implementacija je ostvarena postupno: od jednostavnih funkcionalnosti, dodavajući dio po dio, potpuno je ostvareno raspoređivanje. Na kraju, testiranje je pokazalo da je implementacija uspješna, tj. raspoređivanje zadataka radi onako kako bi i trebalo.

Ključne riječi: raspoređivanje, EDF, period, trenutak krajnjeg završetka, alarm, kontrolne zastavice, dretva, OSIER.

Implementing EDF scheduling for OSIER

Abstract

OSIER is the operating system intended for educational purposes to show the principles of designing components of the operating system for embedded computers. The task of the thesis was to implement the EDF scheduler, and test it on several examples. Implementation is achieved gradually: first only simple functionality is achieved, and then adding piece by piece, full EDF scheduling is achieved. Finally, testing has shown that the implementation is successful, i.e., scheduling tasks work the way it should for EDF.

Keywords: scheduling, EDF, period, deadline, alarm, control flags, thread, OSIER.

Dodatak A
Primjer 3 - Tablica testiranja

Tablica A.1: Tablica raspoređivanja za primjer 3

t [s]	opis	aktivna	edf.ready	edf.wait
0	DOLAZAK D1	–	D1	–
0	DOLAZAK D2	–	D1 D2	–
0	DOLAZAK D3	–	D1 D2 D3	–
0	DOLAZAK D4	–	D1 D2 D3 D4	–
0,6	KRAJ D1	–	D2 D3 D4	D1
1	P_ALARM D1	D2	D3 D4 D1	–
1,5	KRAJ D1	–	D3 D4 D2	D1
1,6	KRAJ D2	–	D3 D4	D1 D2
2	P_ALARM D1	D3	D4 D1	D2
2	P_ALARM D2	D1	D4 D3 D2	–
2,5	KRAJ D1	–	D4 D3 D2	D1
2,6	KRAJ D3	–	D4 D2	D1 D3
3	P_ALARM D1	D2	D4 D1	D3
3	P_ALARM D3	D1	D4 D2 D3	–
3,5	KRAJ D1	–	D4 D2 D3	D1
3,7	KRAJ D2	–	D4 D3	D1 D2
4	P_ALARM D1	D4	D3 D1	D2
4	P_ALARM D2	D4	D3 D1 D2	–
4	P_ALARM D4	D4	D3 D1 D2	–
4	D_ALARM D4	D4	D3 D1 D2	–
4,6	KRAJ D1	–	D3 D2 D4	D1

Nastavlja se na idućoj strani...

Tablica A.1 – Nastavak

t [s]	opis	aktivna	edf.ready	edf.wait
5	P_ALARM D1	D2	D3 D4 D1	–
5,5	KRAJ D1	–	D3 D4 D2	D1
5,6	KRAJ D2	–	D3 D4	D1 D2
6	P_ALARM D1	D3	D4 D1	D2
6	P_ALARM D2	D3	D4 D1 D2	–
6	P_ALARM D3	D3	D4 D1 D2	–
6	D_ALARM D3	D3	D4 D1 D2	–
6,6	KRAJ D1	–	D4 D2 D3	D1
7	P_ALARM D1	D2	D4 D3 D1	–
7,5	KRAJ D1	–	D4 D3 D2	D1
7,6	KRAJ D2	–	D4 D3	D1 D2
7,7	KRAJ D4	–	D3	D1 D2 D4
7,7	KRAJ D3	–	–	D1 D2 D4 D3
8	P_ALARM D1	–	D1	D2 D4 D3
8	P_ALARM D2	D1	D2	D4 D3
8	P_ALARM D4	D1	D2 D4	D3
8,5	KRAJ D1	–	D2 D4	D3 D1
9	KRAJ D2	–	D4	D3 D1 D2
9	P_ALARM D1	D4	D1	D3 D2
9	START D1	D1	D4	D3 D2
9	P_ALARM D3	D1	D4 D3	D2
9,5	KRAJ D1	–	D4 D3	D2 D1
10	P_ALARM D1	D3	D4 D1	D2
10	P_ALARM D2	D1	D4 D3 D2	–
10,5	KRAJ D1	–	D4 D3 D2	D1
11	P_ALARM D1	D2	D4 D3 D1	–
11,5	KRAJ D1	–	D4 D3 D2	D1
11,5	KRAJ D2	–	D4 D3	D1 D2
11,6	KRAJ D3	–	D4	D1 D2 D3
12	P_ALARM D1	D4	D1	D2 D3
12	P_ALARM D2	D4	D1 D2	D3
12	P_ALARM D3	D4	D1 D2 D3	–

Nastavlja se na idućoj strani...

Tablica A.1 – Nastavak

t [s]	opis	aktivna	edf.ready	edf.wait
12	P_ALARM D4	D4	D1 D2 D3	–
12	D_ALARM D4	D4	D1 D2 D3	–
12,6	KRAJ D1	–	D2 D3 D4	D1
13	P_ALARM D1	D2	D3 D4 D1	–
13,5	KRAJ D1	–	D3 D4 D2	D1
13,6	KRAJ D2	–	D3 D4	D1 D2
14	P_ALARM D1	D3	D4 D1	D2
14	P_ALARM D2	D1	D4 D3 D2	–
14,5	KRAJ D1	–	D4 D3 D2	D1
14,6	KRAJ D3	–	D4 D2	D1 D3
15	P_ALARM D1	D2	D4 D1	D3
15	P_ALARM D3	D1	D4 D2 D3	–
15,5	KRAJ D1	–	D4 D2 D3	D1
15,6	KRAJ D2	–	D4 D3	D1 D2
15,7	KRAJ D4	–	D3	D1 D2 D4
16	P_ALARM D1	D3	D1	D2 D4
16	P_ALARM D2	D1	D3 D2	D4
16	P_ALARM D4	D1	D3 D2 D4	–
16,5	KRAJ D1	–	D3 D2 D4	D1
17	KRAJ D2	–	D3 D4	D1 D2
17	P_ALARM D1	D3	D4 D1	D2
17,5	KRAJ D1	–	D4 D3	D2 D1
17,7	KRAJ D3	–	D4	D2 D1 D3
18	P_ALARM D1	D4	D1	D2 D3
18	P_ALARM D2	D1	D4 D2	D3
18	P_ALARM D3	D1	D4 D2 D3	–
18,5	KRAJ D1	–	D4 D2 D3	D1
19	P_ALARM D1	D2	D4 D3 D1	
19,5	KRAJ D1	–	D4 D3 D2	D1
19,5	KRAJ D2	–	D4 D3	D1 D2
19,7	KRAJ D4	–	D3	D1 D2 D4
20	KRAJ TESTIRANJA			