

Operacijski sustavi

prof. Marin Golub

- službena stranica : <http://www.zemris.fer.hr/labosi/os1/tehvel>
- preporučena literatura: Operating System Concepts – Silberschatz

Ivan Nenadić

Predgovor

Ova skripta je rađena po bilješkama sa auditornih predavanja kolegija 'Operacijski sustavi' te je moguće da se u njoj pojavljuju raznorazne greške (tipfeleri, krivi prijepisi s ploče, netočne informacije..) pa molim da sve greške ili gradivo koje nije upisano u ovoj skripti, a trebalo bi biti, prijavite na inenadic@yahoo.com.

Dilanje ove skripte kao i njezino prepravljnje je potpuno besplatno (bar što se autora tiče :-)

P.S. posebna zahvala Zoranu Marinčiću na nadopuni skripte kad ja nisam bio prisutan na predavanjima :-)

Sadržaj

1. Model jednostavnog računala	6
1.1 Dijelovi OS-a	6
1.2 Rudimentalno računalo	6
1.3 Osnovni registri procesora	7
1.4 Procesor trajno radi sljedeće	8
1.5 Osnovni pojmovi	8
1.6 Proces se sastoji od	8
2. Obavljanje I/O operacija, prekidni rad	9
2.1 Prenosjenje znakova radnim čekanjem	9
2.2 Prekidni način rada procesora	10
2.3 Podsustav za prihvrat više prekida	10
2.4 Prenosjenje blokova i sklopovi s neposrednim pristupom	12
3. Datotečni podsustav	13
3.1 Konceptcija datoteka	13
3.2 Opisnik datoteka	13
3.3 Smještaj datoteka u vanjskom spremniku	14
3.4 Tipične datotečne operacije	17
4. Međusobno isključivanje u višedretvenim sustavima	19
4.1 Model višedretvenosti	19
4.2 Ostvarenje međusobnog isključivanja dretvi	21
4.3 Sklopovska potpora međusobnom isključivanju	23
5. Jezgra OS-a	24
5.1 Uvod	24
5.2 Struktura podataka jednostavnog modela jezgre	24
5.3 Prikaz mogućih stanja dretvi	26

5.4 Jezgrine funkcije	26
6. Međudretvena komunikacija	29
6.5 Pokretanje više procesa	29
6.6 Višedretveni rad	29
7. Gospodarenje spremničkim prostorom (Memory Management – MM)	30
7.1 Fragmentacija	30
7.2 Ocjena fenomena fragmentacije (Knuthovo 50% pravilo)	31
7.3 Straničenje	32
7.4 Praktična ostvarenja izbacivanja stranica	33

Slike

Slika 1 - Dijelovi OS-a	6
Slika 2 - procesor i spremnik povezan na sabirnicu	7
Slika 3 - pristupni sklop za prijenos pojedinačnih znakova	9
Slika 4 - više prekidnih signala spojenih na jedan prekidni ulaz procesora	10
Slika 5 - dijagram rada prekida i prekidne rutine	11
Slika 6 - sustav sa sklopom za prihvat prekida	12
Slika 7 - jednostavni model sklopa za neposredni pristup spremniku	12
Slika 8 - Opisnik datoteke	13
Slika 9 - ploče tvrdog diska	14
Slika 10- Fizički sektori nisu logički susjedni sektori	17
Slika 11 - Grafički prikaz 4 strategije posluživanja	18
Slika 12 - Jezgra OS-a	24
Slika 13 – Opisnici	25
Slika 14 - moguća stanja dretvi	26
Slika 15 – statička fragmentacija radnog spremnika	30
Slika 16 - dinamička fragmentacija radnog spremnika	30

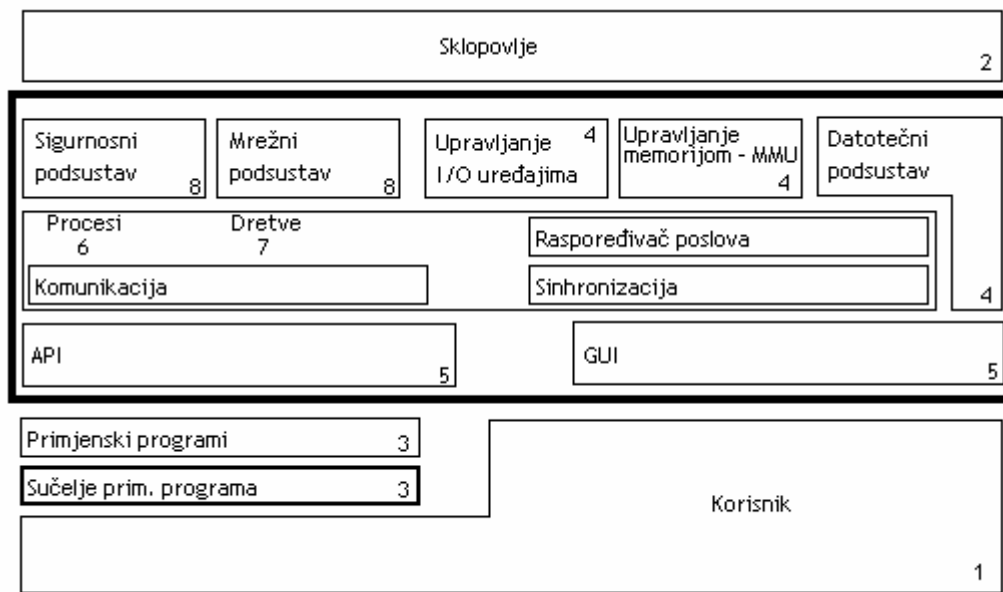
Slika 17 - tipovi blokova	31
---------------------------------	----

1. Model jednostavnog računala

def: Operacijski sustav je skup programa koji nam olakšavaju rad na računalu.

Današnji operacijski sustavi omogućuju **višekorisnički** i **višezadačni** rad.

1.1 Dijelovi OS-a

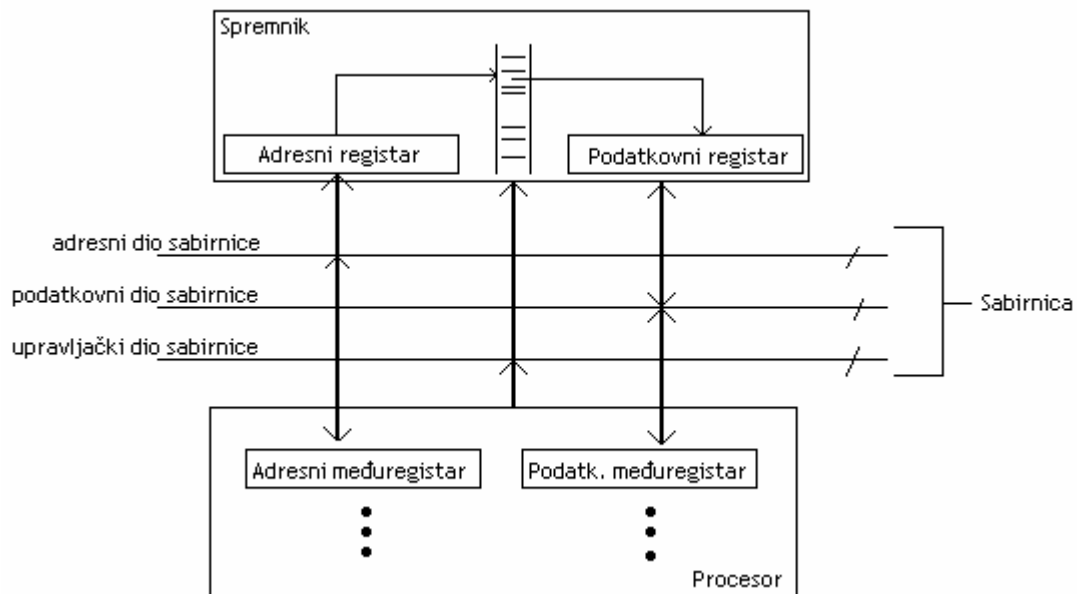


Slika 1 - Dijelovi OS-a

- MMU – Memory Management Unit
- API – Application Program Interface
- GUI – Graphic User Interface
- I/O – Input/Output

1.2 Rudimentalno računalo

- procesor i radni spremnik
- pretpostavke:
 1. U radnom spremniku nalazi se strojni oblik programa i svi potrebni podaci
 2. Rezultati programa također se pohranjuju u radni spremnik
- u jednom sabirničkom ciklusu može se obaviti samo jedno ili čitanje ili pisanje u radni spremnik



Slika 2 - procesor i spremnik povezan na sabirnicu

registri – služe za pohranjivanje svih informacijskih sadržaja koji ulaze ili izlaze iz procesora i u njemu se transformiraju.

1.3 Osnovni registri procesora

- adresni međuregistar (AMR)
- podatkovni međuregistar (PMR)
- instrukcijski međuregistar – pohranjuju se instrukcije koje se izvode
- programsko brojilo (PC)
- registar kazaljke STOG-a (SP – Stack Pointer)
- skup općih registara

1.4 Procesor trajno radi sljedeće

```
Ponavljati {  
    dohvati iz spremnika instrukciju na koju pokazuje PC;  
    PC++;  
    dekodirati instrukciju i odrediti operaciju;  
    odrediti adresu operanada i rezultata;  
    operande dovesti u Aritmetičko-Logičku Jedinicu (ALJ) i izvesti operaciju;  
    pohrani rezultat;  
    Ako je (prekidni signal postavljen) {  
        zabraniti prekidanje;  
        aktivirati sustavni adresni prostor i sustavnu kazaljku STOG-a;  
        PC pohraniti na sustavni STOG;  
        u PC staviti adresu potprograma prekida;  
    }  
}
```

Procesor “provlači” kroz niz instrukcija **instrukcijsku dretvu**.

1.5 Osnovni pojmovi

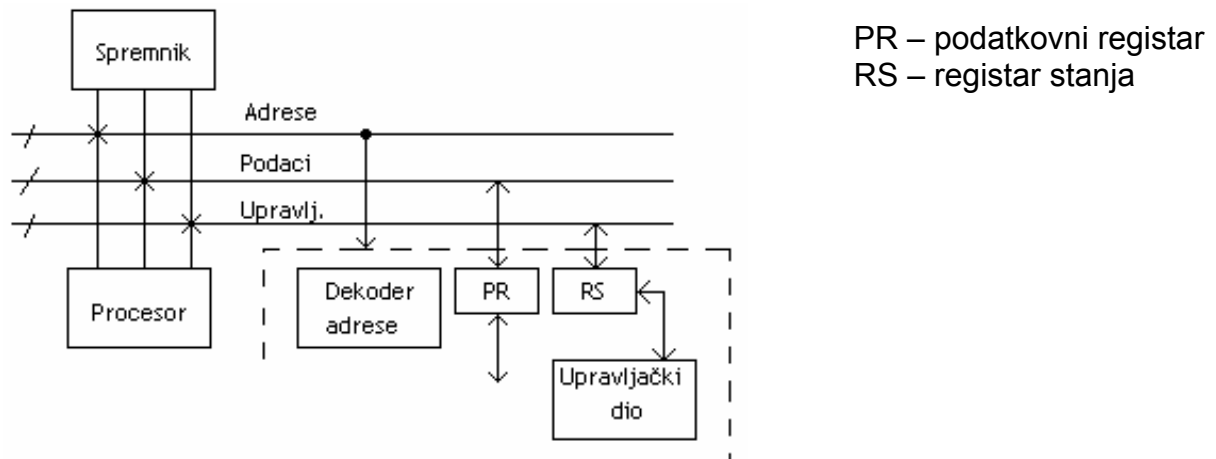
1. Program – nešto što je pohranjeno na papiru, disketi, memoriji,...
 - statični niz instrukcija
2. Dretva (thread) - niz instrukcija koje se izvode
3. Proces
 - skup računalnih resursa koji omogućuju izvođenje programa
 - “sve što je potrebno” da bi se program izvodio
 - okolina koja omogućuje izvođenje programa

1.6 Proces se sastoji od

1. barem jedne dretve
2. zajedničkog adresnog prostora
3. adresnog prostora rezerviranog za pojedinu dretvu
4. STOG, kazaljka STOG-a, opisnici datoteka, opisnici cijevovoda, redovi poruka, semafori...

2. Obavljanje I/O operacija, prekidni rad

2.1 Prenošnje znakova radnim čekanjem



Slika 3 - pristupni sklop za prijenos pojedinačnih znakova

Prenošnje jednog bajta prema izlaznoj napravi:

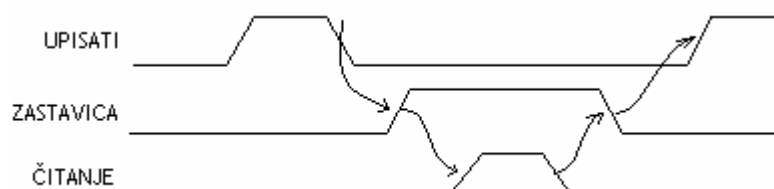
1. Jednom instrukcijom bajt se dobavlja iz spremnika u registar procesora
2. Drugom instrukcijom bajt se prenosi u podatkovni registar

Prihvatanje jednog bajta od ulazne naprave:

Jedan način prijenosa naziva se radno čekanje:

Pročitati RS;
dok je (zastavica == 0) pročitati RS;
pročitati PR; //ovim čitanjem zastavica se vraća u '0'

Dvožično rukovanje:



2.2 Prekidni način rada procesora

ZASTAVICA ↑ - generira električki signal koji se preko posebnog vodiča dovodi do procesora.

PREKID – procesor prelazi iz korisničkog načina rada u sustavski ili jezgri
način rada

- pozivaju se potprogrami koji sačinjavaju jezgru OS-a
- procesor pri prelasku u jezgri način rada djeluje tako da:
 - onemogućiti prekidanje
 - adresira sustavski dio spremnika
 - aktivira sustavski registar sustavske kazaljke STOG-a
 - programsko brojilo ide na sustavski STOG
 - u PC se postavlja kazaljka na početak potprograma
- ponašanje procesora treba nadopuniti tako da on na kraju izvođenja svake instrukcije ispituje da li se pojavljuje prekidni signal (to se na različitim arhitekturama izvodi na različite načine)

Pogledati 'procesor trajno radi sljedeće'

Na početku potprograma sve registre pohraniti na sust. STOG (**pohraniti kontekst**)

Na kraju programa vratiti registre obrnutim redoslijedom (**obnoviti kontekst**)

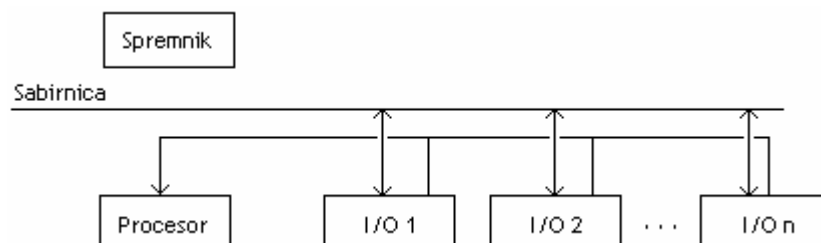
Prije nego što se PC sa sustav. STOG-a vrati, treba se vratiti iz prekidnog načina rada:

- omogućiti prekidanje
- prebaciti adresiranje u korisnički adresni prostor
- aktivirati korisnički registar kazaljke STOG-a

2.3 Podsustav za prihvati više prekida

Najjednostavniji oblik:

(Zastavica 1, zastavica 2,...zastavica n)

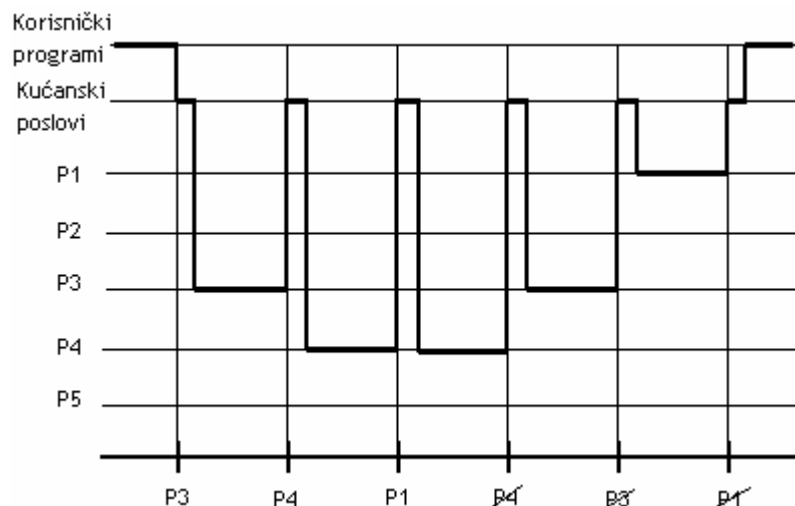


Slika 4 - više prekidnih signala spojenih na jedan prekidni ulaz procesora

Nedostatak: manje važan prekid (čija obrada dugo traje) može ometati rad važnijih prekida

Podsustav za prihvat više prekida razvrstanih po prioritetima s najjednostavnijim sklopovljem:

- navedeni nedostatak može se otkloniti programskim rješenjem
- pristupni sklopovi su razvrstani po važnosti ili prioritetu (npr. veći broj – veći prioritet)



Slika 5 - dijagram rada prekida i prekidne rutine

Prekidna rutina () {

(prekidni signal ↑, prekidanje onemogućeno, PC je na sust. STOG-u)
 pohrani kontekst na sust. STOG;
 ispitati uzrok prekida, indeks prekida i;

Ako je ($1 \leq i \leq N$) {

oznaka_čekanja[i] = 1;

zastavica_i = 0;

Dok (postoji oznaka_čekanja[j] != 0 && j > tekući_prioritet) {

odaberi najveći j;

oznaka_čekanja[j] = 0;

zapis[j] ← kontekst sa sust. STOG-a, tekući_prioritet;

tekući_prioritet = j;

omogući prekidanje;

potprogram za obradu prekida (j);

zabrani prekidanje;

zapis[j] → kontekst na sust. STOG, tekući_prioritet;

} obнови kontekst sa sust. STOG-a;

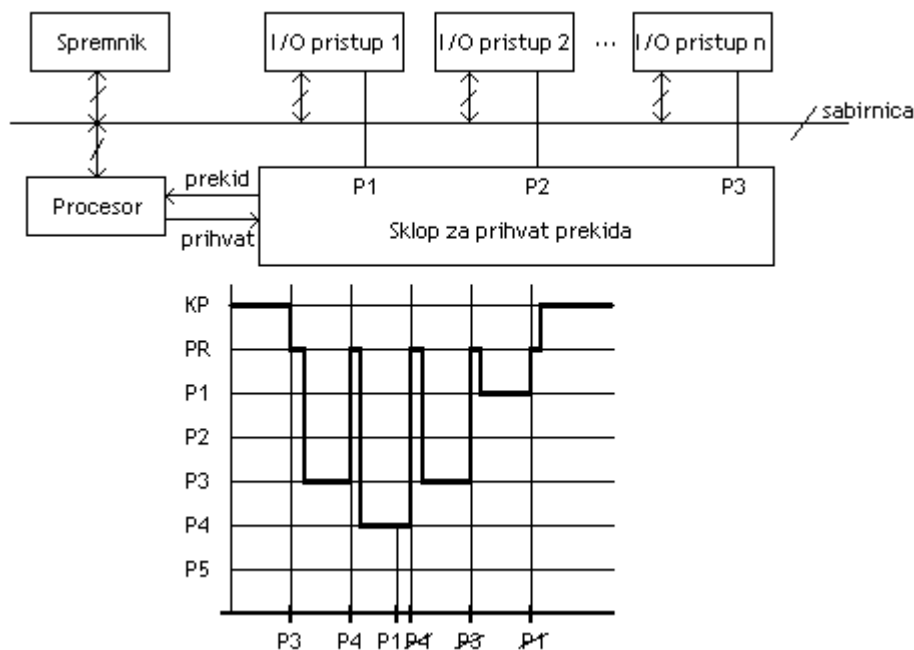
omogući prekidanje;

vрати se iz prekidnog načina rada;

}

}

Sklopovska potpora za ostvarivanje višestrukog prekidanja:



Slika 6 - sustav sa sklopom za prihvatanje prekida

Prekidi generirani unutar procesora:

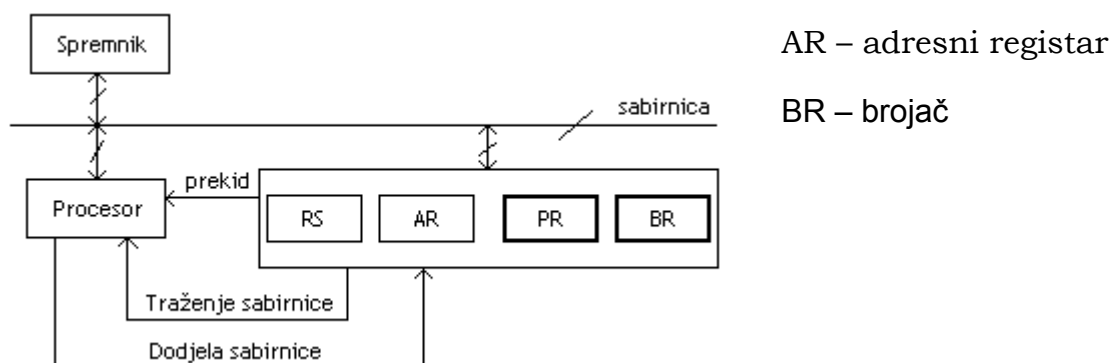
Uzrokuju ih:

- dijeljenje s nulom
- adresiranje nepostojeće lokacije u adresnom prostoru
- dekodiranje nepostojeće instrukcije

Programski prekidi:

Postoji barem jedna instrukcija koja omogućuje namjerno izazivanje prekida iz programa.

2.4 Prenos blokova i sklopovi s neposrednim pristupom



Slika 7 - jednostavni model sklopa za neposredni pristup spremniku

3. Datotečni podsustav

3.1 Konceptcija datoteka

Datoteka je skup informacija povezanih u jednu cjelinu.

Datoteka i rad s datotekama je najvidljiviji dio OS-a.

Datotečni podsustav sastoji se od:

1. datoteka
2. direktorija (tablica)
3. procedura za manipuliranje datotekama

OS sadrži naredbe za rad s direktorijima i datotekama

osnovne naredbe:

- stvori direktorij: **md** ili **mkdir**
- briši direktorij: **rmdir**
- premjesti se: **cd**
- kopiraj: **cp**
- premjesti datoteku ili direktorij: **mov**
- briši: **rm**

3.2 Opisnik datoteka

Svaka datoteka je u potpunosti opisana svojim opisnikom.

ime datoteke
tip datoteke
lozinka
ime vlasnika
pravo pristupa
vrijeme stvaranja
ime zadnjeg korisnika
vrijeme zadnje promjene
opisnik smještaja

pravo pristupa (zastavice): [UNIX]

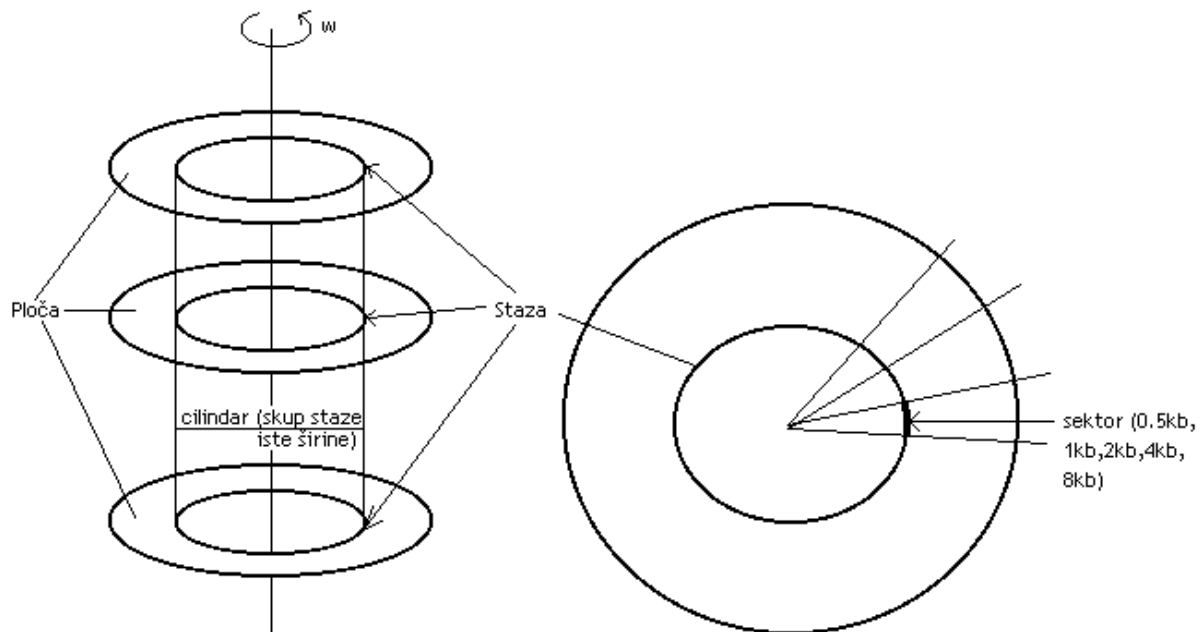
User			Group			Other		
r	w	x	r	w	x	r	w	x

chmod o-rwx .. briše rwx flagove sa 'other'

chmod go+rx .. stavlja flagove rx na 'group' i 'other'

Slika 8 - Opisnik datoteke

3.3 Smještaj datoteka u vanjskom spremniku



Slika 9 - ploče tvrdog diska

Staze su podijeljene na sektore jednake veličine (kapacitet)

Svaki vanjski spremnik ima svoju datotečnu tablicu.

Na disku su neke od staza rezervirane za smještaj tablica.

Organizacija tablica se razlikuje od OS-a do OS-a.

Osnovni sadržaj datotečne tablice je:

1. broj sektora po disku
2. broj slobodnih sektora
3. Informacije o slobodnim sektorima
4. tablica opisnika pohranjenih datoteka

Informacije o slobodnim sektorima – evidencija o slobodnom prostoru na disku:

a) binarni prikaz

011010011010



sektor zauzet

sektor slobodan

- prednost: sektori s greškom se prilikom formatiranja mogu postaviti u 1 (i više ih nikad ne možemo koristiti)
- mana: slobodan prostor saznajemo brojeći nule (traje)

b) lista slobodnih blokova (skupina sektora) X

c) lista slobodnih skupina blokova ✓

Svaki sektor ima jedinstvenu adresu koja se sastoji od:

- a) rednog broja ploče
- b) rednog broja staze na ploči
- c) rednog broja sektora na stazi

Adresira se blok bajtova a ne 1 bajt! (1 sektor ili više)

NTFS (MS WINDOWS NT, 2000, ...) – NT File System

Prostor na disku dodjeljuje se po skupinama sektora (cluster)

Cluster se sastoji od n sektora

n	vel. diska	B (byte)
1	< 512Mb	512
2	do 1Gb	1024
4	do 2Gb	2048
8	> 2Gb	4056

LNC – logički broj skupine

- redni broj skupine (od početka do kraja)
- fizička adresa = $n \cdot \text{LNC}$

MFT – datoteka; glavna tablica datoteka (Master File Table)

- svaka datoteka (pa i MFT) ima u toj datoteci zapis koji je opisnik datoteke
- datoteka MFT je podijeljena na zapise a svaki taj zapis je veličine jedne skupine
- male datoteke smještavaju se unutar MFT zapisa
- velike datoteke – u MFT-u se nalaze indeksi skupina sektora (podaci o smještaju datoteke) i ukoliko je potrebno, MFT zapis se proširuje za dodatni zapis.

VCN – virtualni redni broj skupine (Virtual Cluster Number)...0, 1, 2, ...

Za smještanje datoteke pronalaze se uzastopne skupine sektora (ako ih ima) i dodjeljuju se datoteci.

UNIX FS: i-node (indeksni čvor)

U opisniku se tipično predviđa:

10k – 10 direktnih kazaljki (na sektore ili skupine sektora)

10+255 – 1 jednostruko indirektna kazaljka

Uzimamo da je sektor veličine 1k, kazaljke veličine 1B

266+256·256 – 1 dvostruko indirektna kazaljka

- 1 trostruko indirektna kazaljka

$(10+256+256 \cdot 256+256^3 \approx 16Gb)$

Ukupno trajanje prijenosa podataka

trajanje postavljanja glave

trajanje traženja staze (seek time)

→ ovisi o početnom i krajnjem položaju glave

→ sastoji se od:

1. ubrzavanje glave
2. konstantna brzina
3. usporavanje
4. fino podešavanje

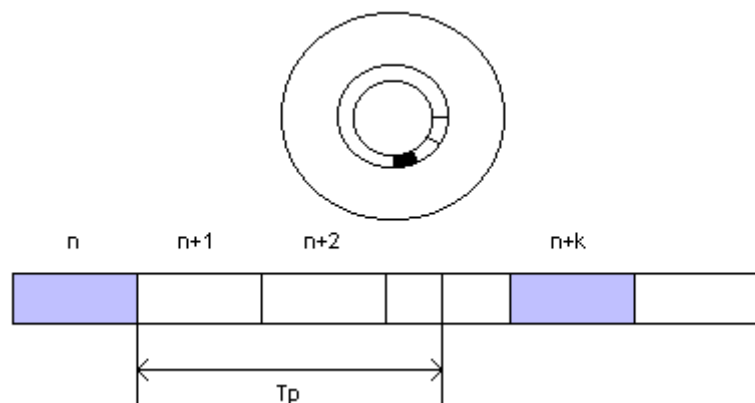
+ rotacijsko kašnjenje = $T_R/2$ T_R – vrijeme jednog okretaja

+ trajanje prijenosa podataka (data transfer time)

trajanje čitanja dijela ili cijele staze

(po potrebi treba uzeti u obzir faktor PREPLITANJA)

+ trajanje premještanja glave sa staze na stazu



Slika 10- Fizički sektori nisu logički susjedni sektori

Ako su dva sektora koji logički slijede jedan iza drugoga fizički smješteni na sektorima n i $n+k$ tada je faktor preplitanja jednak ' k '.

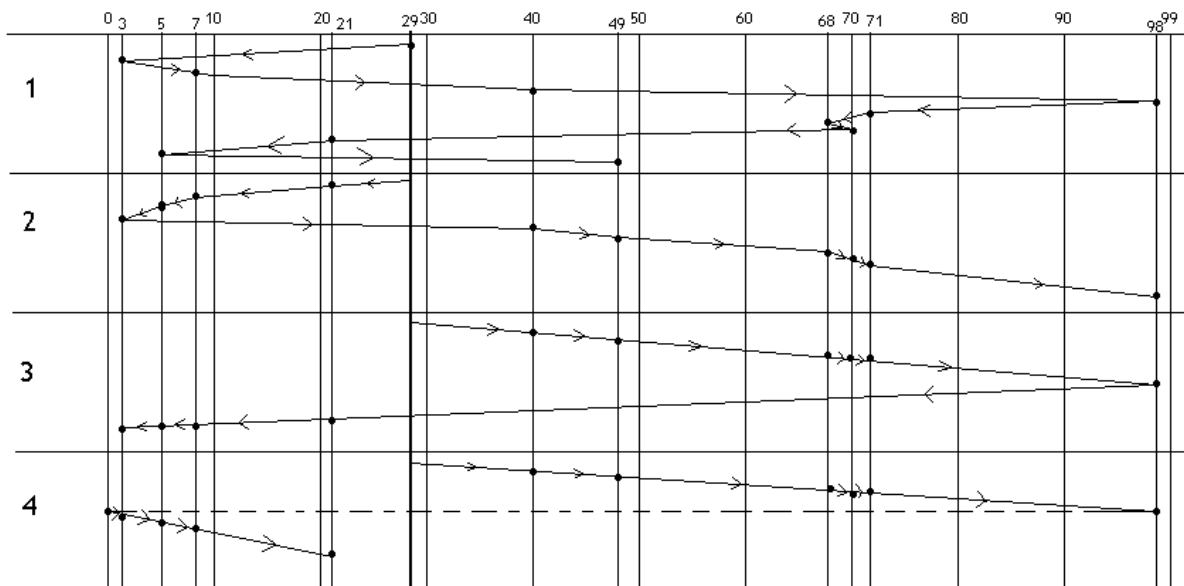
3.4 Tipične datotečne operacije

- stvoriti
- izbrisati
- kopirati
- premjestiti
- otvoriti
- zatvoriti
- čitati
- pisati

Zadatak:

Disk s pokretnim glavama ima 100 staza (0-99). Neka se glava trenutno nalazi na stazi 29 s tim da je prije bila na stazi 8. Zahtjevi za pristup pojedinim stazama svrstani su po redu prispjeća: 3, 7, 40, 98, 71, 68, 70, 21, 5, 49.

- koliki je ukupan pomak glave pri izvršenju tih zahtjeva za sve 4 strategije posluživanja?
- opisati svaku od navedenih strategija
- kod kojih strategija može doći do izgladnjivanja (beskonačnog odgađanja posluživanja nekih zahtjeva zbog posluživanja novopristiglih zahtjeva)?
- grafički prikazati kretanje glave prilikom obrade danih zahtjeva za sve 4 strategije



Slika 11 - Grafički prikaz 4 strategije posluživanja

- FCFS (First Come First Served) – posluživanje po redu prispjeća.
- SSTF (Shortest Seek Time First) – posluživanje najbližih zahtjeva (može doći do izgladnjivanja).
- SCAN – skeniranje (ovisi o položaju glave prije početnog).
- C-SCAN (Circular SCAN) – kružno skeniranje (posluživanje).

4. Međusobno isključivanje u višedretvenim sustavima

4.1 Model višedretvenosti

Dretve

- proces se sastoji od barem jedne dretve
- podaci rezervirani za proces dostupni su svim dretvama
- dretveni dio spremničkog prostora sastoji se od:
 - dijela gdje su smještene instrukcije
 - STOG, kazaljka STOG-a
 - lokalni podaci
- problem sinhronizacije prilikom pristupa zajedničkim podacima treba nekako riješiti
- dretve se mogu izvoditi istodobno (paralelno) na više procesora ili prividno istodobno (prividno paralelno) ako se izvode na istom procesoru (dretve se izvode naizmjenično)

Dretva **T** čita podatke iz domene **D**, obavlja neku funkciju **f(D)** nad podacima iz domene i rezultat upisuje u poddomenu **K**. (**f(D)=K**).

T_i – i-ta dretva, **f_i : D_i → K_i**

DVA ZADATAKA SU MEĐUSOBNO NEZAVISNA AKO NEMAJU ZAJEDNIČKIH LOKACIJA U SVOJIM DOMENAMA I PODDOMENAMA.

Uvijek nezavisnosti zadataka **i** i **j** je:

$$(K_i \cap K_j) \cup (D_i \cap K_j) \cup (D_j \cap K_i) = 0$$

Pišu na isto mjesto
Piše tamo gdje drugi čita

K-kodomena – mjesto gdje se upisuju rezultati

Kada zadaci pišu na isto mjesto, zadaci su zavisni.

~~$(D_i \cap D_j)$~~ – nema jer smijemo čitati iz iste memorijske lokacije

Z_1 – zadatak koji obavlja dretva

$Z_1 < Z_2$ – ukoliko se mora poštivati redoslijed

$T_1 \rightarrow T_2$ – prvo se mora obaviti T_1

Zadatak:

Sustav zadataka je zadan u obliku lanca: $Z_1 \rightarrow Z_2 \rightarrow Z_3 \rightarrow Z_4 \rightarrow Z_5 \rightarrow Z_6 \rightarrow Z_7$

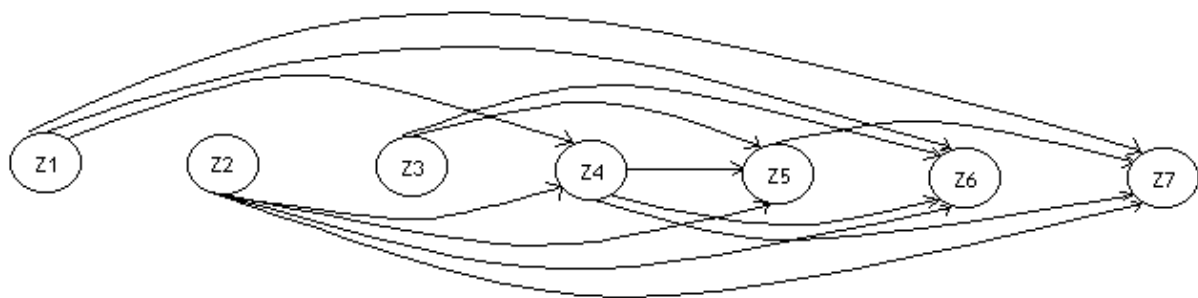
Zadaci imaju domene **D** i kodomene **K** prema tablici:

	Z_2	Z_3	Z_4	Z_5	Z_6	Z_7	Z_8
M1	D	D	D			K	
M2	K			K		D	D
M3		K		D	K		K
M4			K		D		

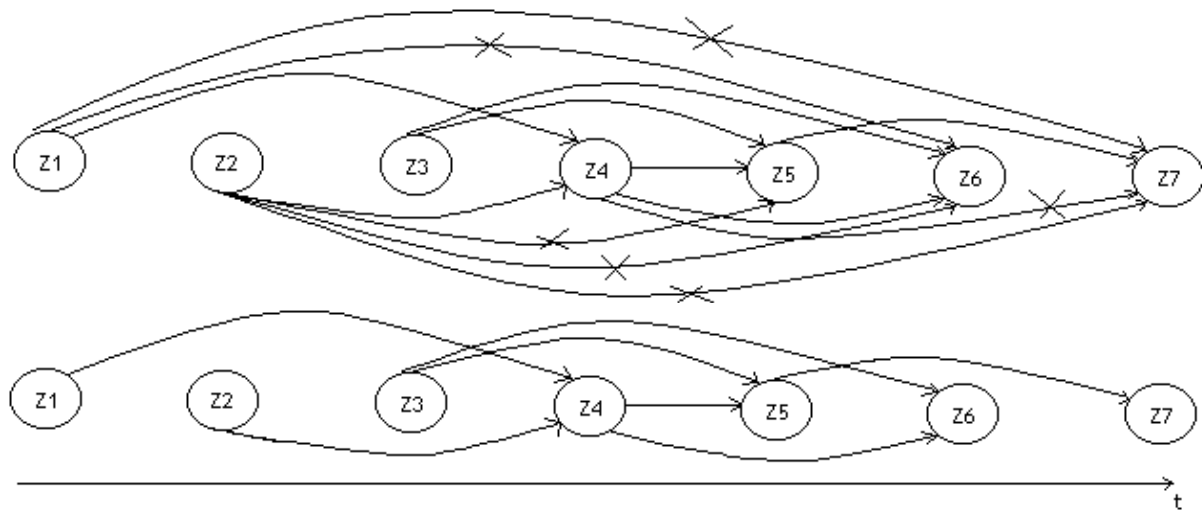
*M1-M4 – Skupine memorijskih lokacija

Odrediti maksimalni paralelni sustav zadataka uzimajući u obzir njihov međusobni odnos u lancu.

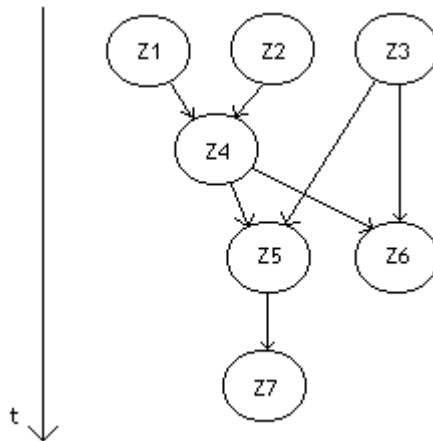
1. Utvrđivanje zavisnosti $(K_i \cap K_j) \cup (D_i \cap K_j) \cup (D_j \cap K_i) = 0$



2. Eliminacija redundantnih (suvišnih) zavisnosti:



3. Određivanje maksimalnog paralelnog sustava zadatka:



4.2 Ostvarenje međusobnog isključivanja dretvi

```

Dretva {
    dok je (1) {
        K.O. // Kritični odsječak (critical section)
        N.K.O. // Ne-Kritični odsječak
    }
}

```

Kritični odsječak je dio programa u kojem dretve koriste neko zajedničko sredstvo (zajednička memorija, datoteka..).

ALGORITAM MEĐUSOBNOG ISKLJUČIVANJA MORA ZADOVOLJITI SLJEDEĆA 4 UVJETA:

Samo jedna dretva u nekom trenutku može biti u K.O.

Algoritam međusobnog isključivanja mora djelovati i onda kada su brzine izvođenja dretvi različite.

Ako neka dretva zastane u N.K.O. to ne smije spriječiti drugu dretvu da uđe u K.O.

Izbor koja dretva treba ući u K.O. mora se obaviti u konačnom vremenu.

```
Dretva {
    dok je (1) {
        dok je (ZASTAVICA==1); //radno čekanje
        ZASTAVICA=1;
        K.O.;
        ZASTAVICA=0;
        N.K.O.;
    }
}

dok je (1) {
    dok je (PRAVO!=I);
    K.O.;
    PRAVO=J;
    N.K.O.;
}
```

Rješenje:

Dekkerov algoritam (služi za međusobno isključivanje dvije dretve (Petersonov algoritam također)).

```
dok je (1) {
    ZASTAVICA[ I ]=1;
    dok je (ZASTAVICA[ J ]==1) {
        ako je (PRAVO==J) {
            ZASTAVICA[ I ]=0;
            dok je (PRAVO==J);
            ZASTAVICA[ I ]=1;
        }
    }
    K.O.;
    PRAVO=J;
    ZASTAVICA[ I ]=0;
    N.K.O.;
}
```

Lamportov algoritam (za međusobno isključivanje n dretvi)

4.3 Sklopovska potpora međusobnom isključivanju

1.TAS (Test And Set) – nedjeljiva instrukcija

```
dok je (1) {  
    dok je (TAS_zastavica==1);  
    K.O.;  
    ZASTAVICA=0;  
    N.K.O.;  
}
```

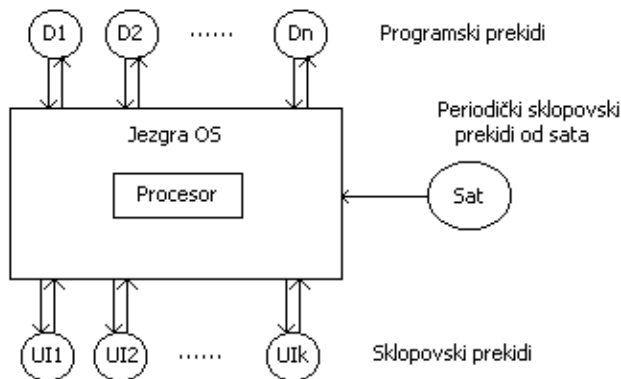
2.SWAP

```
dok je (1) {  
    a=1;  
    dok je (a==1) SWAP(KLJUČ,A);  
    K.O.;  
    KLJUČ=0;  
    N.K.O.;  
}
```

3.FETCH_AND_ADD

5. Jezgra OS-a

5.1 Uvod



PREKID izaziva ulazak u jezgru.

JEZGRA odlučuje kad će se koja dretva obavljati.

Izlaz iz jezgre – jezgrina funkcija je završila, nastavlja se izvoditi prekinuta dretva.

Slika 12 - Jezgra OS-a

Jezgra se sastoji od:

1. struktura podataka
2. jezgrinih funkcija

5.2 Struktura podataka jednostavnog modela jezgre

Opisnik dretve sastoji se od:

kazaljke

PID (Process ID)

ID dretve

stanja dretve (pasivna, aktivna, *blokirana*, pripravna)

prioriteta

početne adrese dretvenog spremničkog prostora

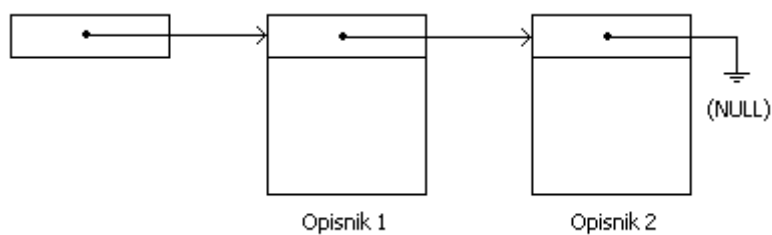
veličine dretvenog spremničkog prostora

adrese prve instrukcije

kašnjenja

prostora za smještaj konteksta (sadržaj svih registara)

Opisnici dretvi su smješteni u listama, tj. redcima.



Slika 13 – Opisnici

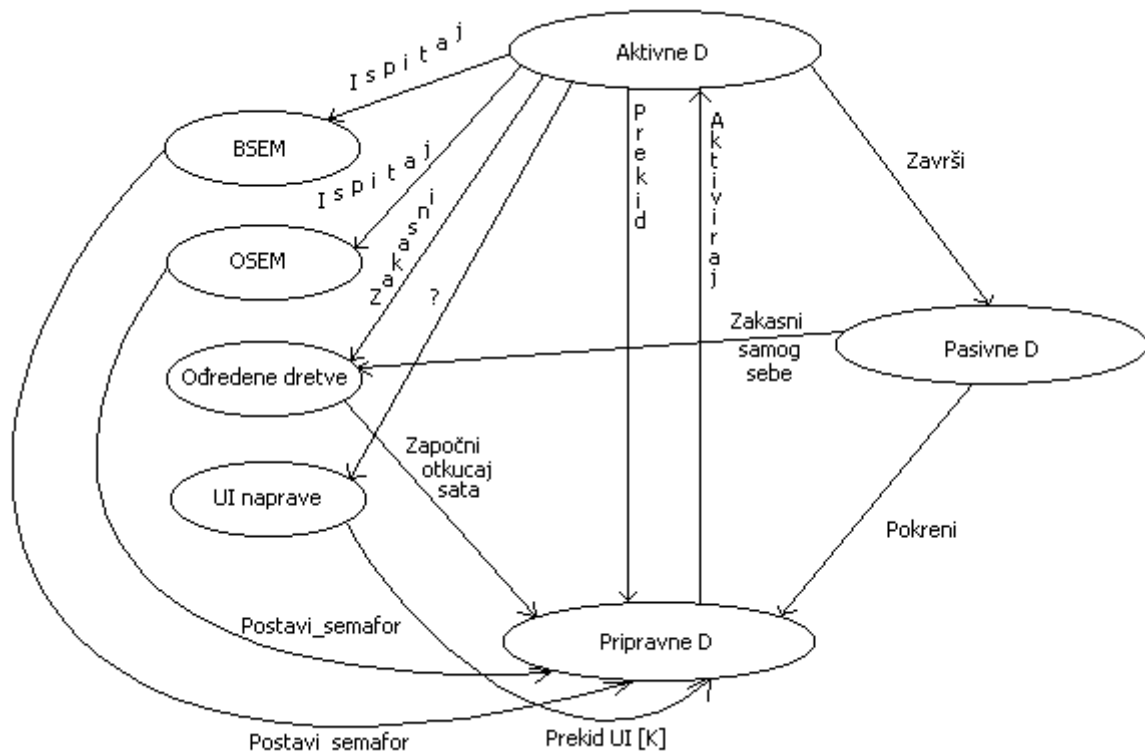
Red – po redu prispjeća

- prioritetni red

Liste ili redovi:

- lista postojećih dretvi
- lista aktivnih dretvi
- lista pripravnih dretvi
- red semafora (binarni – BSEM, opći – OS, OSEM)
- red odgođenih dretvi
- red UI naprave

5.3 Prikaz mogućih stanja dretvi



Slika 14 - moguća stanja dretvi

5.4 Jezgrine funkcije

a) Ulazak u jezgru (ili poziv jezgrine funkcije) zbiva se pod utjecajem prekida kada se dogđa sljedeće:

- onemogućiti prekidanje
- pospremi programsko brojilo na sustavski stog (da se 'program' zna vratiti tamo gdje je prekinut)
- pohrani kontekst na sustavski stog
- poziva se odgovarajuća jezgrina funkcija

Na početku svake jezgrine funkcije treba pohraniti kontekst sa sustavskog stoga u opisnik 'aktivna dretva', to se zove *pohraniti kontekst u opisnik aktivna_D*.

b) Izlaz iz jezgre zbiva se kada jezgrina funkcija završi, svodi se na aktiviranje jedne od dretvi (aktivira prvu dretvu iz reda pripravnih dretvi), a događa se sljedeće:

- premjestiti prvi opisnik iz reda pripradne_D u red aktivna_D
- obnoviti kontekst opisnika aktivna_D
- omogućiti prekidanje

- vratiti se iz prekidnog načina rada(PC sa sustavskog stoga u PC registar, PC se zadnji obnavlja)

Usporedba binarnog i općeg semafora:

Binarni semafor		Opći semafor
BSEM		OS
Semafor je jezgryn mehanizam koji omogućuje međusobno isključivanje dretvi, a sastoji se od:		
BSEM.v 0 → neprolazan (crveno) 1 → prolazan (zeleno)	← jedne varijable → jedne kazaljke	OS.v <=0 → neprolazan >0 → prolazan
Red semafora obično se formira po redu prispjeća.		
čekaj_BSEM postavi_BSEM	Moguća stanja su : <ul style="list-style-type: none"> • <i>semafor je prolazan</i> • <i>neprolazan ali je red prazan</i> • <i>neprolazan ali netko čeka</i> 	čekaj_OS postavi_OS

Što se događa prilikom poziva jezgrinih funkcija (semafora)?

	zeleno	crveno (nitko ne čeka)	crveno (netko čeka)
čekaj_BSEM	$v = 0$ (prođi)	pređi u red BSEM	
postavi_BSEM	- (već je zeleno)	$v=1$ (nema nikog)	premjesti prvi opisnik iz reda BSEM u red priprave dretve(aktivira prvu dretvu koja se nalazi u redu)
čekaj_OS	$v--$; (prođi)	$v--$; (pređi u red OS, ide u red koji hoće proći preko semafora)	
postavi_OS	$v++$;	$v++$;	$v++$; (ako je vrijednost $== 0$, premjesti prvi opisnik iz reda OS u red priprave dretve)
čekaj_OSEM	$v--$;	prijedi u red OSEM	
postavi_OSEM	$v++$; (crveno je, ako nema nikoga u redu, pozeleni)	$v++$;	premjesti prvi opisnik iz reda OSEM u red priprave dretve (prpustiti prvu dretvu)

OS ima mnogo crvenih i zelenih stanja

OSEM ima mnogo prolaznih (zelenih) stanja, i samo jedno neprolazno

6. Međudretvena komunikacija



"Međudretvena komunikacija.ppt"

vidjeti predavanja power point.....

6.5 Pokretanje više procesa

`int fork (void);`

Vraća:

'-1' ako se dogodi greška ili ako nemože stvoriti novi proces

'0' ako uspije stvoriti proces dijete, ono dobiva vrijednost '0'

vraća PID (process id)

Proces roditelja i proces djeteta nemaju isti PID

Kada se stvara novi proces, OS ne kopira sve varijable u proces dijete, to se dogodi samo onda kad se dotična varijabla koristi u procesu djeteta.

Proces dijete postaje proces 'zombie', ako ga roditelj ne čeka (`wait (NULL)`), kada proces završi on vraća `NULL` procesu roditelj, pa stoga proces roditelj treba čekati dok se proces dijete ne završi.

6.6 Višedretveni rad

`thr_create(...argumenti...);` // stvara više dretvi unutar istog procesa

U sustavu POSIX `pthread_create (.....);`

`thr_create()` vraća: '-1' → ako se dogodi greška ili ako nemože stvoriti novu dretvu

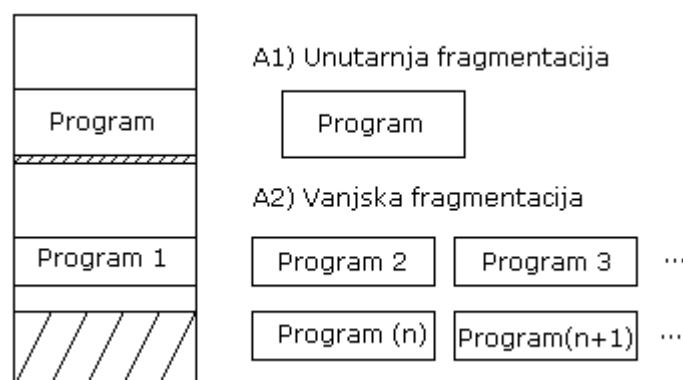
7. Gospodarenje spremničkim prostorom (Memory Management – MM)

7.1 Fragmentacija

Radni spremnik

A) statičko raspoređivanje radnog spremnika

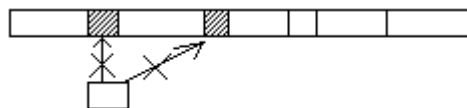
- veličina okvira (stranice) je jednaka
- okvir na disku je obično veličine jednog sektora



Slika 15 – statička fragmentacija radnog spremnika

- 1) Unutarnja fragmentacija – *(moj komentar, ubuduće ovom bojom i italic:-) program koji se smješta u memoriju nikada nije **točno** velik kao i stranica pa će uvijek ostati mali dio praznog prostora u bloku/sektoru.*
- 2) Vanjska fragmentacija – Svi programi koji su smješteni u određenu particiju su blokirani a drugi koji čekaju nisu pripremljeni za izvođenje u toj particiji.

B) dinamičko raspoređivanje radnog spremnika



Slika 16 - dinamička fragmentacija radnog spremnika

Vanjski spremnik (HD,...)

Pojavljaju se statička unutarnja i dinamička fragmentacija (pogledati primjere gore).

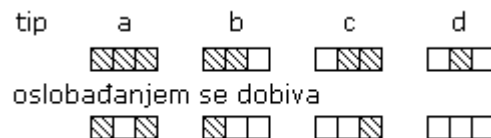
Unutarnja fragmentacija kod velike datoteke nije višekratnik veličine slobodnih sektora.

Kolika je unutarnja fragmentacija tvrdog diska?

$$U_F = \text{Broj datoteka} \cdot \text{veličina sektora} / 2$$

7.2 Ocjena fenomena fragmentacije (Knuthovo 50% pravilo)

Postoje 4 tipa blokova (punih):



Slika 17 - tipovi blokova

prvi redak blokova:

broj punih blokova: $m = a + b + c + d$

broj rupa: $n = (2d + b + c) / 2$; $b = c \Rightarrow (2d + 2b) / 2 \Rightarrow n = b + d$

drugi redak blokova:

vjerojatnost da broj rupa povećamo za 1 = vjerojatnost oslobađanja $\cdot a/m$

vjer. da broj rupa smanjimo za 1 = vjer. oslobađanja $\cdot d/m + (\text{vjer. zahtjeva} \cdot q)$
($q < 1$); ($p + q = 1$)

U kojem odnosu su m i n ?

Uvjet ravnoteže:

vjerojatnost zauzimanja = vjerojatnost oslobađanja

vjer. da se broj rupa poveća za jedan = vjer. da se broj rupa smanji za 1

$$\frac{a}{m} = \frac{d}{m} + q \quad a = d + m(1 - p)$$

$$\begin{aligned} m &= a + b + c + d \\ m &= d + m(1-p) + b + c + d, \quad b = c \\ m &= 2d + m - mp + 2b = \\ &= 2(d+b) + m - mp \\ m &= 2n + m - mp \\ 2n &= mp \\ n &= (m / 2) \cdot p; \quad (p \approx 1) \Rightarrow n = m / 2 \end{aligned}$$

U stacionarnom stanju broj rupa jednak je polovici broja punih blokova.

7.3 Straničenje

Strategije izbacivanja stranica

Zadatak: u sustavu sa straničenjem program veličine 400 riječi (1-400) generira slijed adresa: 23, 47, 333, 81, 105, 1, 400, 157, 30, 209, 289, 149 i 360.

Program ima na raspolaganju 200 riječi primarne memorije.

Napisati niz referenciranja stranica veličine 50 riječi (to je veličina stranice).

Koliki je postotak promašaja za sve tri strategije izbacivanja stranica?

Zahtjevi	23	47	333	81	105	1	400	157	30	209	289	149	360
Stranice	1	1	7	2	3	1	8	4	1	5	6	3	8
FIFO	1 - - -		1 7 - -	1 7 2 -	1 7 2 3		8 7 2 3	8 4 2 3	8 4 1 3	8 4 1 5	6 4 1 5	6 3 1 5	6 3 8 5
LRU	1 - - -		1 7 - -	1 7 2 -	1 7 2 3		1 8 2 3	1 8 4 3		1 8 4 5	1 6 4 5	1 6 3 5	8 6 3 5
OPT	1 - - -		1 7 - -	1 7 2 -	1 7 2 3		1 7 8 3	1 4 8 3		5 4 8 3	5 6 8 3		

FIFO (First In First Out) – izbacuje se ona stranica koja je bila prva dohvaćena. Svako dobavljanje stranice iz tvrdog diska u radni spremnik naziva se promašaj, a pogodak kada se traži stranica koja se već nalazi u radnom spremniku.

LRU (Last Recently Used) – izbacuje se ona stranica koja se u povijesti najdulje nije koristila.

OPT (optimalna strategija) – uzima u obzir budućnost i nije jednoznačna.

Broj stranice u koju spada određena riječ dobije se tako da se podijeli broj te riječi sa veličinom stranice..npr. $23 / 50 = 0.xx \Rightarrow$ spada u prvu stranicu, $81 / 50 = 1.xx \Rightarrow$ spada u drugu stranicu.

Postotak promašaja se računa kao broj_promašaja / broj_zah_tjeva..npr. za FIFO: $(11 / 13) \cdot 100\%$


7.4 Praktična ostvarenja izbacivanja stranica

Zamjena se treba obaviti što je moguće brže.

- a) “algoritam što je moguće brže”
- b) “postotak promašaja što je moguće manje”

Registar povijesti (posmačni, shift registar)

1	0	0	0	0	0	0	0	0	1
---	---	---	---	---	---	---	---	---	---



 Otkucaj sata => pomak
 (izbacuje se ona stranica koja ima najmanji broj)

Korištenje registra povijesti u praksi je neizvedivo.

Intel x86

- nema registra povijesti
- postoje samo dvije vrste stranica:
 - 1) stranice kojima se u prethodnoj periodi otkucaja sata pristupalo
 - 2) one stranice kojima se nije pristupalo

bit čistoće (ako je postavljen => stranica je čista, u nju se nije ništa zapisivalo).

Pristup	Čistoća
0	0
0	1
1	0
1	1



Više UNIX operacijskih sustava i WinNT (i viši) koriste inačicu LRU strategije koja se naziva SATNI MEHANIZAM.

- sve stranice u radnom spremniku razvrstane su u FIFO listu (trivijalno za izvesti)
- lista se obilazi kazaljkom koja kada dođe do kraja, vraća se na početak
- izbacuje se onaj okvir (ako je to potrebno) na koji pokazuje kazaljka i bit pristupa mu je jednak nuli a kazaljka se pomiče za jedno mjesto u desno
- ako je bit pristupa '1' tada se on resetira i kazaljka se pomiče za jedno mjesto u desno

Moguće izvesti samo na jednoprocesorskom sustavu!

PITANJA ZA 1. KOLOKVIJ IZ OPERACIJSKIH SUSTAVA (Informatički odjel)

Uvod; Model jednostavnog računala

1. Skicirati procesor i spremnik povezane na sabirnicu. Sabirnicu podijeliti na tri dijela.
2. Nabrojiti osnovne registre procesora.
3. Opisati osnovne operacije procesora pri izvođenju jedne instrukcije.
 - a) bez mogućnosti prekidanja
 - b) s mogućnošću prekidanja
4. Objasniti pojmove program, proces i dretva.

Obavljanje U/I operacija, prekidni rad

5. Opisati osnovne komponente pristupnog sklopa (prepoznavatelj adrese, registre...).
6. Objasniti pojam "radno čekanje".
7. Skicirati signale "dvožičnog rukovanja" između pristupnog sklopa i ulazno-izlazne naprave.
8. Opisati što se događa kada se pojavi prekid.
9. Kako djeluje procesor pri prelasku u jezgrin način rada?
10. Napisati u pseudokodu što procesor trajno radi i kako treba nadopuniti ponašanje procesora tako da se omogući prekidni način rada?
11. Objasniti pojmove "pohraniti kontekst", "obnoviti kontekst" i "vratiti se iz prekidnog načina rada".
12. Čemu služi i kako djeluje sklop za prihvatanje prekida?
13. Što sve uzrokuje prekid generiran unutar procesora?
14. Zadatak: "U sustavu s jednostavnim sklopovljem javljaju se prekidi...".
15. Nabrojiti registre sklopa za neposredni pristup spremniku i objasniti njihovu funkciju.

Međusobno isključivanje u višedretvenim sustavima

16. Objasniti pojmove višezadačni rad, višedretveni rad i višekorisnički rad.
17. Što je to dretva i od čega se sastoji dretveni spremnički prostor? Na koji način dretve istog procesa međusobno najlakše komuniciraju?
18. Naveći i obrazložiti uvjet nezavisnosti zadataka.
19. Zadatak: "Sustav zadataka je zadan u obliku lanca...".
20. Nabrojiti uvjete koje treba zadovoljavati algoritam međusobnog isključivanja.
21. Čemu služi Dekkerov, a čemu Lamportov algoritam?
22. Naveći instrukcije procesora koje predstavljaju sklopovsku potporu međusobnom isključivanju. Za barem jednu od njih objasniti kako se koristi u međusobnom isključivanju.

Jezgra operacijskog sustava

23. Naveći najznačajnije sadržaje opisnika dretvi.
24. Opisati strukturu podataka jednostavnog modela jezgre.
25. Koja su moguća stanja dretvi?
26. Nacrtati graf prijelaza dretvi iz stanja u stanje.
27. Što se događa prilikom poziva jezgrine funkcije?
28. Što se sve zbiva kada jezgrina funkcija završi?
29. Opisati način ostvarenja i osnovnu upotrebu binarnog semafora.
30. Opisati način ostvarenja i osnovnu upotrebu općeg semafora.
31. Čemu služi semafor i u čemu se razlikuju binarni i opći semafor?
32. Kako treba proširiti jezgrine funkcije u višeprocorskom sustavu?

1.

2. Nabrojiti osnovne registre procesora.

1. adresni međuregistar (za dohvat odnosno spremanje podataka)
2. podatkovni međuregistar (za dohvat odnosno spremanje podataka)
3. instrukcijski registar (spremaju se instrukcije kad se dobave)
4. programsko brojilo ili PC (za dohvat instrukcija, pokazuje na sljedeću instrukciju)
5. registar kazaljke stoga (stek pointer)
6. registar stanja (registar zastavica)

3. Opisati osnovne operacije procesora pri izvođenju jedne instrukcije.

1. dohvatiti iz spremnika instrukciju na koju pokazuje programsko brojilo
 2. povećati PC za jedan (povećati PC tako da pokazuje na sljedeću instrukciju)
 3. dekodirati instrukciju i odrediti operaciju koju treba izvesti
 4. odrediti adresu operanda i rezultata
 5. operande dovesti u AL (aritmetičko-logičku jedinicu) i izvesti zadanu operaciju
 6. pohraniti rezultat u neki opći registar
- to se ponavlja sve dok je procesor uključen

4. Objasniti pojmove: program, dretva i proces.

PROGRAM - statični niz instrukcija
- nešto što je pohranjeno na papiru, disketi, memoriji itd.

DRETVA - niz instrukcija u izvođenju, dijeli se na niti

PROCES - je program koji se izvodi ili program u izvođenju zajedno sa svim potrebnim resursima

Sastoji se od:

- barem jedne dretve
- zajedničkog adresnog prostora
- adresnog prostora rezerviranog za svaku pojedinu dretvu
- ostalo – stog, kazaljka stoga, opisnici datoteka, opisnici cjevovoda, redovi poruka, semafori, uvjetne varijable, varijable zaključavanja itd.

5. Opisati osnovne komponente pristupnog sklopa.

- sklop za dekodiranje adrese (dobavlja adresu podatka)
- podatkovni registar (za podatke)
- registar stanja (daje procesoru stanje u kojem se nalazi naprava)
- upravljački dio (određuje stanje naprave)

6. Što je to radno čekanje?

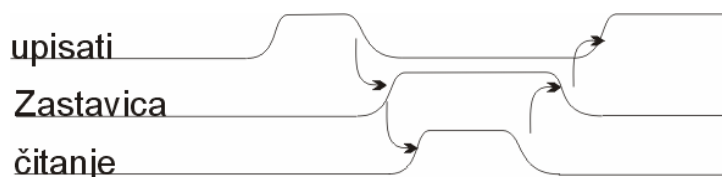
RADNO ČEKANJE je prijenos znakova gdje:

1. procesor treba pročitati registar stanja
2. dok je odgovarajuća zastavica u registru stanja ZASTAVICA=0 nemoj raditi ništa nego pročitaj ponovo RS (registar stanja), tek kad je ZASTAVICA=1 može se pročitati podatkovni registar (PR).

SKRAČENO:

pročitaj (RS);
dok je (ZASTAVICA==0) pročitaj (RS);
pročitaj (PR);

7. Skicirati signale «dvožilnog rukovanja» između pristupnog sklopa i U/I naprave.



8. Opisati što se događa kada se pojavi prekid.

Kada se dogodi prekid procesor prelazi iz KORISNIČKOG u JEZGRIN (sustavski) način rada. Nakon toga pozivaju se podprogrami koji čine jezgru OS-a.

9. Kako djeluje procesor pri prelasku u jezgrin način rada?

Procesor zabrani daljnje prekidanje, adresira sustavski dio spremnika, aktivira sustavski registar kazaljke stoga. Programsko brojilo sprema na sustavski stog, te se u PC na početak podprograma sprema kazaljka.

10. Napisati u pseudokodu što procesor trajno radi i kako treba nadopuniti ponašanje procesora tako da se omogući prekidni način rada?

ponavljati {

.
.
.

//na kraju dolazi

ako je (prekidni signal postavljen) {
 zabrani prekidanje;
 aktivirati sustavski adresni prostor i sustavsku kazaljku stoga;
 PC ↔ sustavski stog;
 PC ↔ adresa početka podprograma;
}

} sve dok je (procesor uključen);

11. Objasniti pojmove: pohraniti kontekst, obnoviti kontekst i vratiti se iz prekidnog načina rada.

POHRANITI KONTEKST - na početku podprograma treba spremiti sve registre na sustavski stog, a taj čin se naziva pohraniti kontekst (kontekst je skup svih registara procesora).

OBNOVITI KONTEKST - na kraju podprograma treba vratiti registre obrnutim redoslijedom

VRATITI SE IZ PREKIDNOG NAČINA RADA – prije no što se PC (programsko brojilo) sa sustavskog stoga vrati u PC (sadržaj se vrati) treba:

1. omogućiti prekidanje
2. prebaciti adresiranje u korisnički adresni prostor

aktivirati korisnički registar kazaljke stoga

12. Čemu služi i kako djeluje sklop za prihvatanje prekida? ***

Sklop za prihvatanje prekida služi za filtriranje prekida tako da kad se dogodi prekid manjeg prioriteta sklop ne šalje prekid procesoru (ne ometa ga kao softversko ostvarenje ovog sklopa, tako da procesor ni ne zna da je bio prekid)

Umjesto zastavice sklop ima niz prekidnih vodiča od pojedinih U/I sklopova pa tek onda u procesor.

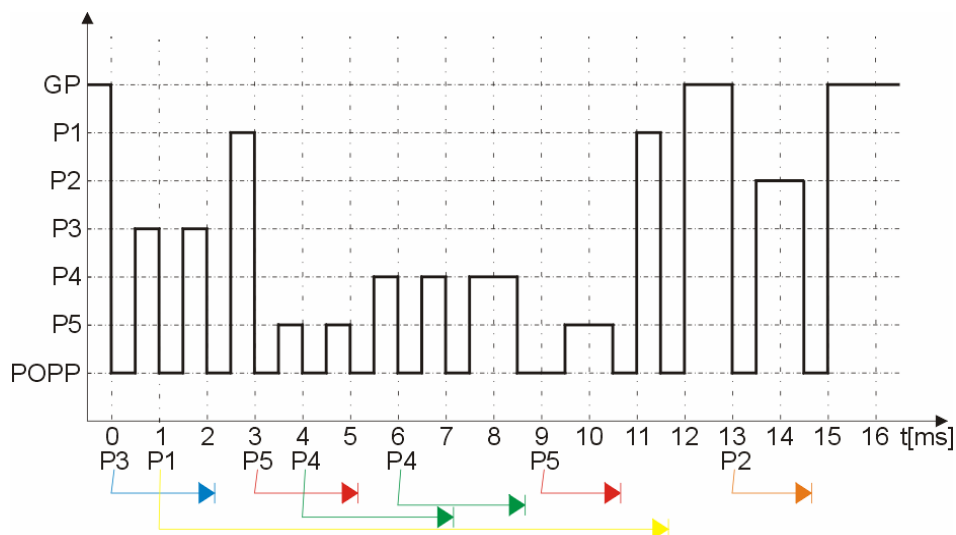
13. Što sve uzrokuje prekid generiran unutar procesora?

- djeljenje s nulom
- adresiranje nepostojeće lokacije u adresnom prostoru dretve
- dekodiranje nepostojećeg instrukcijskog koda

14. Zadatak sa predavanja (dijagram prikazuje prekidnog rada)

U sustavu s minimalnim sklopovljem javljaju su prekidi P1, P2, P3, P4 i P5. Prekidi P1, P2 i P3 javljaju se u trenucima 1ms, 13ms, 0ms, respektivno (istim redoslijedom). Prekid P4 javlja se dva puta u trenucima 4ms i 6ms. Prekid P5 javlja se dva puta u trenucima 3ms i 9ms. Prioritet prekida određen je brojem (P5 ima najviši prioritet). Svaka od prekidnih rutina traje 1ms.

Grafički prikazati aktivnost procesora u glavnom programu (GP), procedurama za obradu prekida (Pi) i proceduri za određivanje prioriteta prekida (POPP) koja traje 0.5ms.



15. Nabrojiti registre sklopa za neposredni pristup spremniku i pojasniti njihovu funkciju.

- ADRESNI REGISTAR (AR) - sprema se početna adresa bloka koji se želi prenesti
- BROJAČ (BR) - sprema se broj znakova koji se prenose
- REGISTAR STANJA (RS) - daje procesoru zahtjev za prekid ***
- PODATKOVNI REGISTAR (PR) - spremaju se podaci koji se prenose ***

16. Objasniti pojmove višezadačni rad, višedretveni rad i višekorisnički rad

Višezadačni rad (multitasking) – mogućnost računala da obavlja istovremeno više poslova

Višedretveni rad (multithreading) – brzo obavljanje više procesa u jednom programu

Višekorisnički rad (multiuser system) – računalo koje mogu koristiti više osoba

- svima je zajedničko da obavljaju više zadataka odjednom

17. Što je to dretva i od čega se sastoji dretveni spremnički prostor? Na koji način dretve istog procesa međusobno najlakše komuniciraju?

Dretva je proces koji je dio nekog većeg procesa ili programa.

Dretveni spremnički prostor se sastoji od: niza instrukcija (program)
stoga i kazaljke stoga
lokalnih podataka

18. **Navesti i obrazložiti uvjet nezavisnosti zadataka.**

$$(D_i \cap K_j) \cup (D_j \cap K_i) \cup (K_i \cap K_j) = \emptyset$$

Kada za dvije dretve vrijedi uvjet tada su ta dva zadatka nezavisna. Ako su bilo koji od ova tri uvjeta ispunjeni zadaci su zavisni.

$D_i \cap K_j$ – u isto vrijeme ita dretva čita podatak i jeta dretva zapisuje – ako se to dogodi onda su dretve zavisne

$D_j \cap K_i$ – u isto vrijeme jeta dretva čita podatak i ita dretva zapisuje – ako se to dogodi onda su dretve zavisne

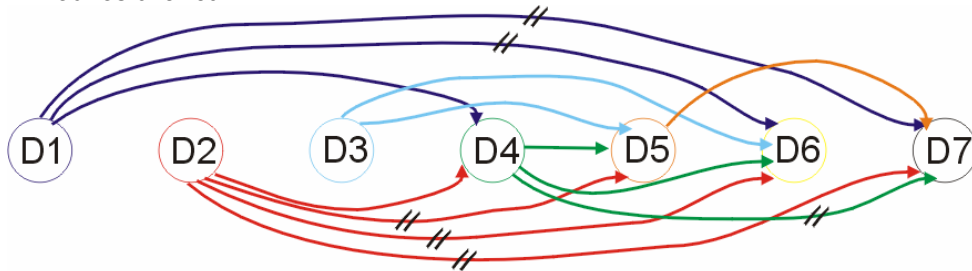
$K_i \cap K_j$ – i ita i jeta dretva zapisuju na istu kodomenu – ako se to dogodi onda su dretve zavisne

19. **Zadatak sa predavanja (sustav zadataka)**

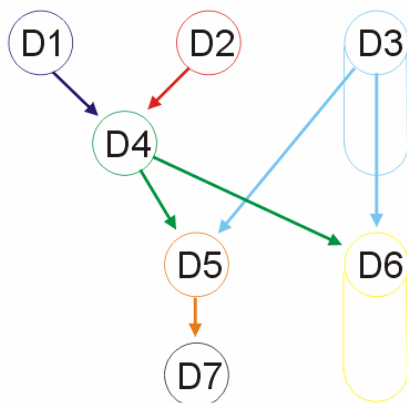
Sustav zadataka je zadan u obliku lanca: $Z_1 \rightarrow Z_2 \rightarrow Z_3 \rightarrow Z_4 \rightarrow Z_5 \rightarrow Z_6 \rightarrow Z_7$ a zadaci imaju domene (D) i kodomene (K) prema tablici:

	Z1	Z2	Z3	Z4	Z5	Z6	Z7
M1	D	D	D			K	
M2	K			K		D	D
M3		K	K	D	K		K
M4			K		D		

Odrediti maksimalno paralelni sustav zadataka uzimajući u obzir njihov međusobni odnos u lancu.



Rješenje:



D1, D2 i D3 se mogu izvršavati paralelno

D4 i D3 se mogu paralelno

D5 i D6 idu paralelno

D6 i D7 mogu paralelno

20. Nabrojiti uvjete koje treba zadovoljavati algoritam međusobnog isključivanja.

1. Dvije dretve ne smiju obavljati K.O. (kritični odsječak)
2. Mehanizam mora djelovati kako brzo se obavljaju dretve
3. Ako neka dretva bude u N.K.O. ne smije spriječiti drugu da uđe u K.O.
4. U konačnom vremenu mora se odlučiti koja će ući prva

21. Čemu služi Dekkerov, a čemu Lamparov algoritam?

Dekkerov postupak služi za međusobno isključivanje dvije dretve.

Lamparov postupak služi za međusobno isključivanje više dretvi.

22. Navesti instrukcije procesora koje predstavljaju sklopovsku potporu međusobnom isključivanju. Za barem jednu od njih objasniti kako se koristi u međusobnom isključivanju.

TAS (test & set)

SWAP (zamijeni)

FETCH_AND_ADD – dohvati i povećaj za 1

dok je (TAS(varijabla)!=0); → radi ništa

TS ispita neku varijablu i postavi je

23. Navesti najznačajnije sadržaje opisnika dretvi. ***

OPISNIK (DESKRIPTOR) DRETVE – je zapis u kojem su pohranjeni podaci o dretvi i sastoji se od:

1. kazaljke ili više njih
2. identifikacijski broj procesa kojoj dretva pripada (PID)
3. identifikacijski broj dretve (ID)
4. stanje dretve može biti
 - pasivna
 - aktivna
 - blokirana
 - pripravna (za izvođenje)
5. prioritet (mjesto gdje je zapisan prioritet)
6. početna adresa dretvenog spremnika prostora
7. veličina dretvenog spremničkog prostora
8. adresa prve instrukcije
9. zadano kašnjenje
10. prostor za smještanje konteksta

24. Opisati strukturu podataka jednostavnog modela jezgre ***

1. LISTA (RED) POSTOJEĆIH DRETVI – kada se dretva nalazi na samo jednoj listi onda je u pasivnom stanju

2. LISTA (RED) AKTIVNIH DRETVI

- koje se izvode, broj članova u toj listi je jednak broju procesora
- u jednoprocesorskom sustavu ima samo jedan deskriptor

3. LISTA (RED) PRIPRAVNIH DRETVI

- ako se ne izvode (spremne su) onda se nalaze na ovoj listi
- prema načinu formiranja red može biti:
 - a) po redu prispjeća (FCFS ili FIFO)
 - b) prioritetni

4. RED SEMAFORA – dretve tijekom izvođenja mogu biti blokirane čekajući na ispunjenje nekog uvjeta

4 načina blokiranja i to čekanjem na:

- BSEM (binarni semafor)
- OSEM (opći semafor)
- istek zadanog intervala kašnjenja
- završetak U/I operacije

5. LISTA (RED) ODGOĐENIH DRETVI – (Mi – zadano kašnjenje i te dretve)

6. LISTA (RED) U/I NAPRAVE – takvih redova ima onoliko koliko ima U/I naprava

- U/I naprave se obično koriste pojedinačno

25. Koja su moguća stanja dretvi?

1. Aktivna dretva
2. Pasivne dretve
3. BSEM (binarni semafor)
4. OSEM (opći semafor)
5. Odgođene dretve
6. U/I
7. Pripravne dretve

26. Nacrtati graf prijelaza dretvi iz stanje u stanje.

27. Što se događa prilikom poziva jezgrine funkcije?

Onemogućuje se daljnje prekidanje → ne mogu se pozivati druge funkcije. Pohranjuje se kontekst, pamti se gdje je bio prekid. Poziva se odgovarajuća funkcija koja sa sustavnog stoga pohranjuje kontekst u opisnik aktivna d.

28. Što se sve zbiva kada jezgrina funkcija završi?

Aktivira se prva dretva iz reda Pripravne D. u red Aktivna D. Obnovi se kontekst iz opisnika Aktivna D. Omogućuje se prekidanje, vraća se iz prekidnog načina. PC iz konteksta → PC. Sustavski prelazi u korisnički način rada.

29. Opisati način ostvarenja i osnovnu upotrebu binarnog semafora.

Binarni semafor je jezgrin mehanizam koji služi za međusobno isključivanje dretvi, tj. za zaštitu kritičkih odsječaka. On propušta jednu po jednu dretvu.

30. Opisati način ostvarenja i osnovnu upotrebu općeg semafora.

31. Čemu služi semafor i u čemu se razlikuju binarni i opći semafor?

Binarni se od općeg razlikuje samo u realizaciji. Stanja su ista.

32. Kako treba proširiti jezgrine funkcije u višeprocorskom sustavu?

Jezgrine funkcije treba proširiti radnim čekanjem.

```
j-fja... () {  
    dok je (TAS OGRADA_JEZGRE==1)  
        ne radi ništa
```

? - Nisam našao odgovor

* - nisam siguran

Datotečni podsustav

1. Navesti elemente datotečnog podsustava. (18)
2. Navesti barem pet osnovnih naredbi operacijskog sustava za rad s datotekama. (18)
3. Opisati osnovni sadržaj opisnika datoteke. (19)
4. Navesti osnovni sadržaj datotečne tablice. (20)
5. Opisati način evidencije binarnim prikazom o slobodnom prostoru na disku. (20)
6. Koliko iznosi ukupno trajanje prijenosa podataka tvrdi disk ↔ radni spremnik? (21)
7. Navesti tipične datotečne operacije. (21)
8. Opisati organizaciju smještaja sadržaja na magnetskom disku (cilindi, staze, sektori). (19) ?
9. Opisati strategije posluživanja zahtjeva za pristup stazama tvrdog diska. (23) ?
10. Zadatak: " Disk s pokretnim glavama ima 100 staza (0-99). Neka...". (23)

Međudretvena komunikacija

11. Što se događa prilikom poziva funkcije *fork*?
12. Napisati program u C-u ili u pseudokodu koji pokreće dva procesa.
13. Nabrojati mehanizme za komunikaciju među procesima.
14. Napisati program u C-u ili u pseudokodu koji pokreće dvije dretve.
15. Nabrojati mehanizme za sinkronizaciju rada dretvi, odnosno procesa.
16. Što je to potpuni zastoj? ?
17. Navesti nužne uvjete za nastajanje potpunog zastoja.

Gospodarenje spremničkim prostorom

18. Navest vrste fragmenatcije prilikom statičkog i dinamičkog dodjeljivanja spremnika.
19. Navesti i pokazati Knuthovo 50%-tno pravilo.
20. Navesti osnovne strategije zamjene stranica.
21. Zadatak: "U sustavu sa straničenjem program veličine 400 riječi (1-400) generira slijed adresa:...". ?*
22. Čemu služi posmačni registar povijesti. Opisati ga. *?
23. Budući da Intel x86 nemaju registra povijesti opisati strategiju izbacivanja stranica. *
24. Što je to satni mehanizam?

Datotečni podsustav

1. Navesti elemente datotečnog podsustava. (18)

- 1.) Datoteka
- 2.) Tablica (direktorija)
- 3.) Procedura za manipuliranje datotekama

2. Navesti barem pet osnovnih naredbi operacijskog sustava za rad s datotekama. (18)

stvari (mkdir, md), briši (rmdir), premjesti (mv), kopiraj (copy, cp), promijeni tekući direktorij (cd)

3. Opisati osnovni sadržaj opisnika datoteke. (19)

Sadrži:

- 1.) ime datoteke
- 2.) tip datoteke
- 3.) lozinku
- 4.) ime vlasnika
- 5.) prava pristupa
- 6.) vrijeme stvaranja
- 7.) vrijeme zadnje promjene
- 8.) ime zadnjeg korisnika
- 9.) opisnik smještaja

4. Navesti osnovni sadržaj datotečne tablice. (20)

- 1.) Broj sektora po disku
- 2.) broj slobodnih sektora
- 3.) informacije o slobodnim sektorima
- 4.) tablica opisnika pohranjenih datoteka

5. Opisati način evidencije binarnim prikazom o slobodnom prostoru na disku. (20)

binarni prikaz: 011000101001100... ima onoliko 1 i 0 koliko ima sektora, a svaki sektor ima svoju jedinstvenu adresu koja se sastoji od:

- 1.) rednog broja ploče
- 2.) rednog broja staze na ploči
- 3.) rednog broja sektora na stazi

6. Koliko iznosi ukupno trajanje prijenosa podataka tvrdi disk ↔ radni spremnik? (21)

Ukupno trajanje prijenosa podataka =

—+— Trajanje postavljanja glave (head position time)

—+— Trajanje traženja staze (seek time) =

Ovisi o početnom i konačnom položaju glave i sastoji se od:

- 1.) Ubrzavanja ručice glave
- 2.) Konstantne brzine
- 3.) Kočenje, usporavanje
- 4.) Fino pozicioniranje

+ Rotacijsko kašnjenje (rotation latency) = $TR/2$

+ Trajanje prijenosa podataka (rotation transfer time) =

—+— Trajanja čitanja dijela ili cijele staze

+ (faktor preplitanja [interleave faktor])

—+— Trajanje premještanja glave sa staze na stazu

7. Navesti tipične datotečne operacije. (21)

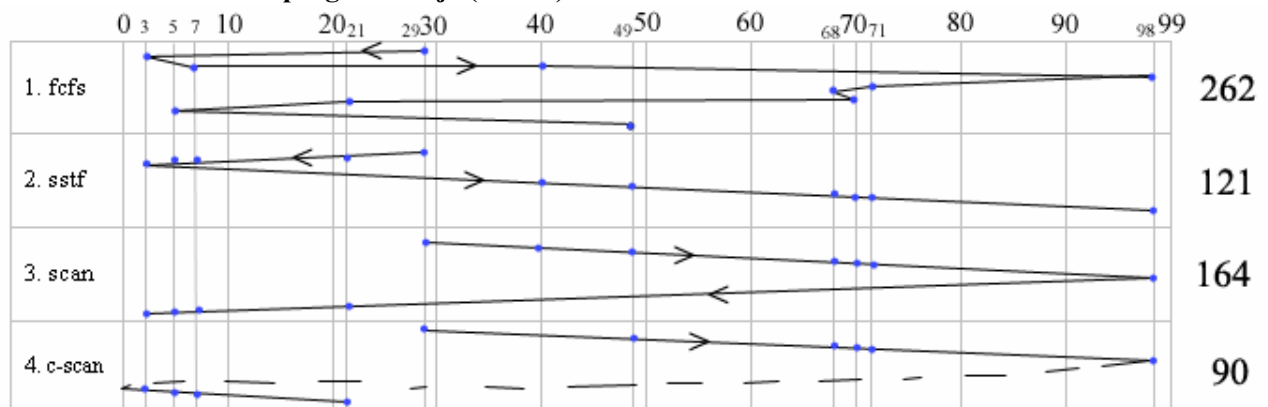
stvoriti, izbrisati, kopirati, premjestiti, otvoriti (file *fopen ("ime datoteke", tip^{r, w ...} datoteke), zatvoriti, čitati (fgets), pisati (fputs) "—————" → pokazivač na opisnik tablice

8. Opisati organizaciju smještaja sadržaja na magnetskom disku (cilindi, staze, sektori). (19)

9. Opisati strategije posluživanja zahtjeva za pristup stazama tvrdog diska. (23)

10. Zadatak: Disk s pokretnim glavama ima 100 staza (0-99). Neka se glava trenutno nalazi na stazi 29 s tim da je prije bila na stazi 8. Zahtjevi za pristup pojedinim stazama, svrstani su po redu prispjeća. Zahtjevi su slijedeći: 3, 7, 40, 98, 71, 68, 70, 21, 5 i 49. Koliki je ukupan pomak glave pri izvršenju tih zahtjeva za strategiju posluživanja:

1. redom prispjeća FCFS (first come first serve)
2. s najkraćim vremenom traženja SSTF (shortest seek time first)
3. pregledavanje (scan)
4. kružno pregledavanje (c-scan)



FCFS – krenem od 29 pa na 3, pa na 40... Cijelo vrijeme pratiš zahtjeve (3,7,40,98,71...).

Kasnije računam ukupan pomak (29-3) + ... = 262

SSTF – gledam od 29 kaj mi je najbliže – 40 ili 21. Sa 21 gledam kaj mi je bliže 7 ili 40. 7...

Kada dođem do kraja, idem natrag na 29 i desno. Na kraju se računa pomak glave.

SCAN – sve zahtjeve na desno do kraja i onda na lijevo.

C-SCAN – opet sve zahtjeve na desno do kraja, onda crtkano do 0, pa ponovo na desno

Međudretvena komunikacija

11. Što se događa prilikom poziva funkcije *fork*?

Sistemske pozivom *fork* zahtijeva se stvaranje novog procesa. Kada proces koji se trenutno izvodi pokrene novi proces, pokrenuti proces postaje proces "dijete" procesa "roditelja" koji ga je pokrenuo. Dijete dobija kopije segmenta instrukcija i segmenta podataka od roditelja. Također, dijete nasljeđuje većinu sistemskih podataka od roditelja.

12. Napisati program u C-u ili u pseudokodu koji pokreće dva procesa.

To si isfurajte u bilježnicu iz OS (labosa) koja je dozvoljena na ispitu. Program ide ovako:

```
#include <stdio.h>
#define br.procesa 2
#define br. prolaza 5
void proces (int i) {
    int j;
    for (j=1 ; j<br. prolaza; j++) {
```

```

        printf("\n Proces: %d, prolaz: %d\n", i, j);
        sleep (1);
    }
}

int main (void) {
    int i;
    for (i=0; i<br. procesa; i++) {
        switch(fork() ) {
            case -1: printf(" Ne mogu stvoriti novi proces! \n");
                    exit (0);
            case 0: Proces(i), exit(0);
            default: break;
        }
    }
    for (i=0; i<br. procesa; i++) wait (null);
    return 0;
}

```

13. Nabrojati mehanizme za komunikaciju među procesima.

- 1.) Zajednička memorija
- 2.) Cijevovodi (obični i imenovani)
- 3.) Varijable okoline
- 4.) Red poruka

14. Napisati program u C-u ili u pseudokodu koji pokreće dvije dretve.

To si isfurajte u bilježnicu iz OS (labosa) koja je dozvoljena na ispitu. Program ide ovako:

```

#include <stdio.h>
#include <pthread.h>
#define br.dretvi 2
void *dretva (void *rbr) {
    int i, brd;
    for (i=1; i<6; i++) {
        printf("Ja sam dretva: %d i i= %d\n", (int)rbr, i);
        sleep (1);
    }
}

int main (void) {
    int i, br[br_dretvi] { *t[br.dretvi]
    pthread_t t;
    for (i=0; i<br.dretvi; i++) {
        br[i]=i+1;
        if (pthread_create (&t, null, dretva (int*) & br[i]))
        {
            printf("Ne mogu stvoriti dretvu!\n");
            exit (1);
        }
        while (pthread_join(t, null)!=0);
        return 0;
    }
}

```

15. Nabrojiti mehanizme za sinkronizaciju rada dretvi, odnosno procesa.

- 1.) Semafor (bsem i osem)
- 2.) Monitor
- 3.) Varijable međusobnog isključivanja (mutex-mutex_int, mutex_lock, mutex_unlock)
- 4.) Uvjetne varijable: cond_vari, cond_nt, ...
- 5.) K.O. = Enter Critical Section, Leave Critical Section
- 6.) Redovi poruka
- 7.) Cjevovodi

16. Što je to potpuni zastoj?

17. Navesti nužne uvjete za nastajanje potpunog zastoja.

- 1.) Dretva koristi sredstvo međusobno isključivo
- 2.) Dretva samo otpušta pridjeljeno joj sredstvo kada obavi svoj dio posla (nemože joj ga nitko oduzeti)
- 3.) Dretva drži dodjeljeno joj sredstvo dok čeka na dodjelu dodatnog sredstva

Gospodarenje spremničkim prostorom

18. Navesti vrste fragmenatcije prilikom statičkog i dinamičkog dodjeljivanja spremnika.

Kod statičkog su unutarnja (interna) i vanjska (eksterna) fragmentacija.

Kod dinamičkog je vanjska fragmentacija.

19. Navesti i pokazati Knuthovo 50%-tno pravilo.

4 tipa punih blokova:



Puni blokovi $m=a+b+c+d$ $b=c \rightarrow n=b+d$

Broj rupa $n=(b+c+2d)/2$

Oslobađanje srednjega:



$VJ_broj\ rupa + 1 = a/m * \text{vjerojatnost oslobađanja}$

$VJ_broj\ rupa - 1 = d/m * \text{vjerojatnost oslobađanja} + vj_zahtjeva * q$

Uvjet ravnoteže: jednake vjerojatnosti

$q=1-p$

$\rightarrow a/m = d/m + (1-p) \rightarrow a = d + (1-p)m$

$m = a + b + c + d = d + (1-p)m + b + c + d$

$m = 2d + m - pm + 2b \quad (p \approx 1)$

$pm = 2n$

$n = m/2$

20. Navesti osnovne strategije zamjene stranica.

- 1.) FIFO (first in first out)
- 2.) LRU (least recently used)
- 3.) OPT – optimalna strategija

21. Zadatak: U sustavu sa straničenjem program veličine 400 riječi (1-400) generira slijed adresa: 23, 47, 333, 81, 105, 1, 400, 157, 30, 209, 289, 149, 360. Program ima na raspolaganju 200 riječi primarne memorije. Napisati niz refenciranja stanica veličine 50 riječi. Koliki je postotak promašaja za sve tri strategije izbacivanja stranica?

22. Čemu služi posmačni registar povijesti. Opisati ga.

Koristi se u praktičnim ostvarenjima izbacivanja stranica.

23. Budući da Intel x86 nemaju registra povijesti opisati strategiju izbacivanja stranica.

Izbacuje se stranica kojoj je bit pristupa == 0, a ako je bit == 1, bit čistoće == 0.

24. Što je to satni mehanizam?

Satni mehanizam je jedna inačica algoritma LRV-a koju koriste OS-evi kao što su Unix i Win NT.