Operativni sistemi

6.12.2013

1.Ukratko objasniti razliku između korisničkog i kernel moda računarskog sistema i šta uzrokuje prelazak iz jednog u drugi. U kom modu se izvršava:

a. Sistemski poziv fork()
b. Algoritam za detekciju uzajamnog blokiranja
c. Bi editor
d. Bibliotečka f-ja sin(x)

U korisničkom modu izvršavaju se korisnički programi. U ovom režimu rada nije dozvoljen pristup određenim mem. oblastima kao I izvršavanje određenog skupa mašinskih instrukcija. Kerner OS-a se izvršava u kernel modu a iz koga se može pristupiti zaštićenim mem. oblastima i izvršavati privilegovane mašinske instrukcije. Prelazak iz jednog u drugi režim rada uzrokuje sistemski poziv, prekid i zamka (trap).

a) Kernel mod
b) Kernel mod
c) Korisnički mod
d) Korisnički mod

9.2.2017

1. Ukratko objasniti razliku između korisničkog I kernel moda. U kom modu se izvršava i zašto :

a) drajver uređaja
b) dispečer
c) Firefox browser
d) algoritam planiranja diska

Drajver uređaja – kernel
Dispečer – kernel
Firefox browser – korisnički
Algoritam planiranja diska – kernel

5.2.2015

1. Uporedite strukture OS-a : monolitnu, mikrokernel i slojevitu po sl kriterijumima : performanse, pouzdanost, proširljivost i objasni zašto?

- Monolitna struktura OS je kolekcija procedura. Svaka procedura može pozivati svaku poznajući njen interfejs (skup parametara i rezultat) i svaka može pristupati deljivim podacima i strukturama podataka OS-a. Postoji glavna, servisne i uslužne procedure. Servisne procedure obavljaju sistemske pozive, a uslužne pomažu jednoj ili više servisnih procedura. Po pitanju perfomansi i pouzdanosti dobra, ali mana joj je loša proširljivost.

- Slojevita struktura: OS je organizovan kao hijerarhija slojeva pri čemu svaki sloj OS-a obezbeđuje usluge slojevima iznad i koristi usluge slojeva ispod u hijerarhiji. Sloj OS-a je implementacija apstraktnog objekta koji enkapsulira podatke i sadrži operacije za manipulaciju tim podacima. Dobro proširljiva zbog modularnosti, međutim loše performanse jer pri sistemskom pozivu svaki sloj

modifikuje parametre, obrađuje podatke i poziva f-je nižih slojeva generišući dodatan rad i usporavajući izvršenje sistemskog poziva. Takođe loša pouzdanost.

- Mikrokernel struktura : Struktuiranje OS-a uklanjanjem neznačajnih komponenti iz kernela i njihovo implementiranje kod sistemskih ili korisničkih programa.Osnovne f-je kernela : kreiranje i uništenje procesa, planiranje procesa, upravljanje memorijom itd. Ostale f-je OS-a implementirane su kao servisi koji se izvršavaju u korisničkom modu. Dobra proširljivost iz razloga što dodavanje novih servisa ne zahteva modifikaciju kernela, dobra pouzdanost (svi serverski procesi izvršavaju se u korisničkom modu i nemaju direktan pristup hardveru i pad nekog od njih ne uzrokuje pad OS-a. Drajver uređaja bi u ovom slučaju bio implementiran kao serverski process koji se izvršava u modu kernela.

27.6.2014

1. Ukratko opisati razliku između korisničkog i kernel moda i zašto je važna za funkcionisanje OS-a. Koja je razlika između sistemskog poziva i prekida?

Sistemski poziv se obavlja u okviru programa napisanog u programskom jeziku visokog nivoa pozivom f-je iz standardne biblioteke uključene u prevodiocu. Nastanak prekida uzrokuju moduli (U/I, memorija) iz razloga kao što su : program, tajmer, U/I, otkaz hardvera. Prekidi se primarno obezbeđuju kao način poboljšanja iskorišćenosti procesora. Prekidi imaju veću fleksibilnost u predaji i preuzimanju upravljanja od korisničkih programa

23.4.2014

1.Koja je razlika između sistemskog poziva I bibliotečke f-je (npr read) sistemski poziv i fscanf (bibliotečka f-ja). Objasniti ovu razliku u načinu izvršavanja sistemskog poziva na primeru write sistemskog poziva za čitanje podataka sa diska.

27.6.2014

1. Prekid se javlja kao reakcija na eksterno asinhrone događaje, dok sistemski poziv predstavlja zahtev od strane procesa koji se izvršava za korišćenje servisa OS-a.

3.02.2017

2. Šta predstavlja kontekst izvršavanja procesa, u kojim situacijama ga treba sačuvati i gde. Da li se u ovim instrukcijama vrši zamena procesa i kako? Objasniti

a) proces je pozvao f-ju signal() nad condition promenljivom
b)proces izvršava while naredbu u čijem uslovu je test and set konstrukcija nad promenljivom čiji je br 1
c) upravo je završen prenos bloka podataka sa diska koji je process zahtevao naredbom read
d) windows process se izvršavao u foreground-u kad se desio prekid pritiskom taster na tastaturi

Kontekst izvršavanja procesa čine podaci u registrima procesora za vreme izvršavanja procesa (registri opšte namene, upravljački I statusni registri, PSW..)

Kontekst izvršavanja procesa se čuva prilikom zamene(komutiranja) procesa koja je uzrokovana prekidom, trap instrukcijom ili sistemskim pozivom u okviru njegovog PCB-a (upravljačkog bloka procesa).
a)

b)
c) Vrši se zamena procesa. Proces prelazi u stanje spreman ili u stanje spreman/suspendovan ukoliko je prethodno bio u stanju suspendovan obzirom da se javio na koji je čekao završetak U/I operacije
d)

5. 12.2013

2. Šta je zamena (komutiranje)procesa i koji deo slike procesa se menja prilikom zamene procesa I kako? OS održava sl stanja procesa… U kom stanju se nalazi process I u koje stanje prelazi nakon ovih događaja:

a) Procesu P1 je istekao dodeljenji vremenski kvant
b) Desio se prekid jedinice diska, završeno je čitanje sa diska koje je zahtevao proces P2
c) Proces P3 je pozvao operaciju semSignal(s) pri vrednosti semafora S=-1.
d) Proces P4 je swap-ovan na disk dok je čekao na završetak read() f-je za čitanje sa diska
e) Proces P5 izabran od strane dispečera

Zamena (komutiranje) procesa predstavlja zamenu trenutno aktivnog procesa (process u stanju izvršavanja) od strane OS-a. Deo slike procesa koje se menja prilikom zamene procesa je PCB odnosno atributi procesa. Menja se stanje procesa u okviru PCB-a

a) Proces P1 se nalazio u stanju izvršavanja i prelazi u stanje spreman
b) Proces P2 se nalazio u stanju blokiran I prezi u stanje spreman
c) Proces P3 se nalazio u stanju izvršavanje I prelazi u stanje blokiran
d) Proces P4 se nalazio u stanje blokiran I prelazi u stanje suspendovan
e) Proces P5 se nalazio u stanju spreman I prelazi u stanje izvršavanje.

23.4.2014

1. **Koja je razlika između stanja procesa spreman i blokiran, a koja između stanja procesa blokiran i blokiran/suspendovan i koji događaji uzrokuju prelazak procesa iz jednog u drugo stanje i obrnuto u oba slučaja. U kom je stanju i u koje prelazi proces nakon što**
   a) **Pozove semWait operaciju semafora P koji ima vrednst 1**
   b) **Nastane prekid kloka kojim je završen dodeljeni vremenski kvant**
   c) **Startuje sistemski poziv receive() za prijem poruke od drugog procesa**

Proces u stanju spreman je spreman za izvršavanje i čeka da ga dispečer odabere za izvršavanje na procesoru. Proces u stanju blokiran je zaustavio neki resurs zbog koga mora da čeka i ne može da se izvršava na procesoru dok ne nestane taj događaj nakon čega prelazi u stanje spreman.Proces koji je u stanju blokiran prelazi u stanje suspendovan/blokiran kada se prebaci(swap-uje) na disk.

a) Izvršavanje -> izvršavanje
b) Izvršavanje -> spreman
c) Izvršavanje -> blokiran

6.2.2015

2.

```
int main()
{
        int p=5;
        pthread-t t1;
        pthread_mutex_t sem;
        pthread_mutex_init(&sem,NULL);
        pthread_create(&t1, NULL, &proba, NULL);
        pthread_mutex_lock(&sem);
        p=p+5;
        pthread_mutex_unlock(&sem);
        pthread_mutex_destroy(&sem,NULL);
        return 1;
}
void * proba (void * arg)
{
        pthread_mutex_lock(&sem);
        p=p-2;
        pthread_mutex_lock(&sem);
        return NULL;
}
```

6.12.2013

3.Navesti prednosti implementacije niti na nivou jezgra (KLT) u odnosu na implementaciju niti na nivou korisnika. U kojim slojevima se može očekivati da će se višenitni proces brže izvršavati od jednonitnog na istom računarskom sistemu za implementaciju niti ULT?

- Kernel može planirati više niti istog procesa na više procesora
- Blokiranje se vrši na nivou niti- kada se nit blokira, jezgro bira spremnu nit istog ili nekog drugog procesa
- Ako se sve niti procesa izvršavaju na jednom procesoru
- Nemaju blokirajući sistemski poziv

23.4.2014

3. TCB(upravljački blok niti) sadrži atribute niti kao što su : stanje niti, prioritet, programski brojač, statičku memoriju za lokalno promenljive niti, za smeštanje lokalnih promenljiva, pokazivač magacina i registri procesora.

Niti istog procesa dele adresni prostor tog procesa(globalne promenljive, otvarane datoteke..), a svaka nit ima sopstveni magacin, stanje, prioritet

Procesi	Niti u okviru procesa
P1	T_{11},T_{12},T_{13}
P2	T_{21}
P3	T_{31},T_{32},T_{33}

Za ULT : T_{21},T_{31},T_{32},T_{33}

Za KLT: T_{11},T_{13},T_{21},T_{31},T_{32},T_{33}

6.2.2015

3. Na izlazu procesa u slučaju KLT-a se generiše: ax,by,cx,dy,ex,fy,gx,hy,ix,jy
Threadx pročita karakter pa se blokira, pa zatim Thready pročita karakter pa se blokira. U slučaju ULT-a izlaz koji se generiše je: ax,bx,cx,dx,ex,fx,gx,hx,ix,jx. Nit Threadx se izvršava samostalno jer kod ULT, prilikom blokiranja niti blokira se ceo proces.

6.12.2013.

5. Kada algoritam bezbednosti detektuje da je sistem u nebezbednom stanju to znači da će doći do uzajamnog blokiranja. Najpre se vrši simulacija dodele resursa, a nakon toga provera da li je takvo stanje dodele resursa bezbedno. Algoritam bezbednosti izbacuje iz skupa svih procesa bezbedne procese nakon što se uveri da je vektor dodatnih zahteva procesa Pi manji ili jednak od treutno raspoloživog resursa. Ako je na kraju skup svih procesa prazan, sistem je u bezbednom stanju, inače nije.
R=(10,15) V=(1,2) (M,N)

	A		C	
Proces	M	N	M	N
P1	2	3	10	5
P2	3	3	9	7
P3	2	2	3	4
P4	2	5	4	8

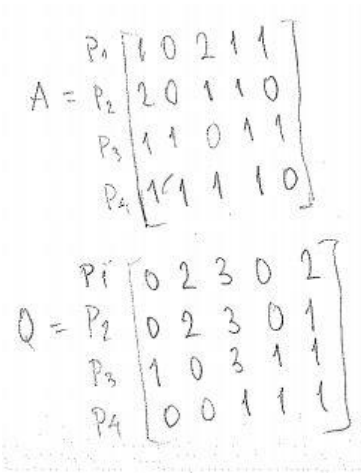
1. W=(1,2)
2. Proces P3 -> Q3<= W
3. W = (1+2)+A3=(3,4)
4. Rest = {P1,P2,P4}
5. Process P4 -> Q4 <= W
6. W=(3,4)+A4=(5,9)
7. Rest {P1,P2}

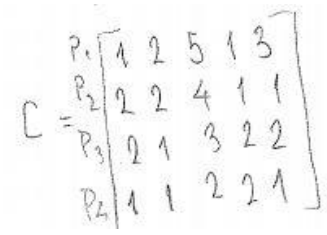
Q=C-A= P1 |8 2|
P2 |6 4|
P3 |1 2|
P4 |2 3|

P1 i P2 ne ispunjavaju uslov Q1<=W(Q2<=w) tako da ostaju u skupu procesa Rest, a pošto skup nije prazan , sistem je u nebezbednom stanju

9.02.2017

5. Strategije sprečavanja i izbegavanja u b ograničavaju pristup resursima i nameću ograničenja procesima tako da su neefikasne.





Najmanja vrednost x je 2 da bi stanje bilo sigurno

6.2.2015
5.

R=150 V=25

Proces	Max	Drži
1	70	45
2	60	40
3	60	15
4	60	25
4	60	35

C= P1 |70|
P2 |60|
P3 |60|

Q= P1 |25|
P2 |20|
P3 |45|
P4 |35|
P4 |25|

A= P1 |45|
P2 |40|
P3 |15|
P4 |25|
P4 |35|

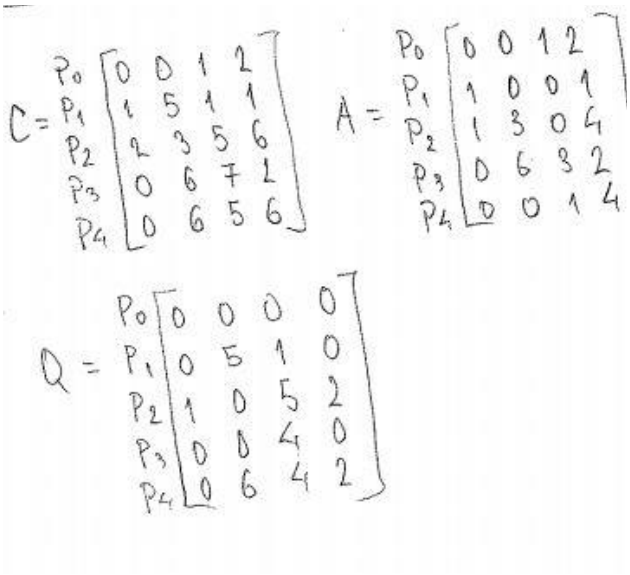
1.W=25
2.P1->Q1<=W
3. W=25+A1=70
4.

Ukoliko se startuje P4 inicijalnim potrebama od 25 memorijskih jedinica sistem će biti u bezbednom stanju, a ako P4 bude 35 onda neće.

23.4.2014
5. Sprečavanje u b predstavlja onemogućavanje nastanka nekog od četiri potrebna uslova za u b .
Postoje dve klase metoda za sprečavanje u b:
Indirektne metode – sprečavanje pojave uzajamnog isključivanja,uslova "drži i čekaj" ili uslova bez prekidanja
Direktne metode – sprečavanje pojave kružnog čekanja

V(1,5,2,0) R(3,14,7,13) V(R1,R2,R3,R4)

1. W=1,5,2,0
2. P0->Q0<=W
3. W=(1,5,2,0)+A0=(1,5,3,2)
4. P1->Q1<=W
5. W=(1,5,3,0)+A1=(2,5,3,3)
6. System nije u sigurnom stanju



9.2.2017

6.

Fizička memorija = 2^{34} B
Maksimalan broj segmenata = 64K
Maksimalna veličina segmenta = 2^{20} stranica
Veličina stranica = 2^{12} B

47	31	11	0
Broj segmenata	Broj stranica	Offset	
64KB	1MB	4KB	

a) Koliko bitova ima logička (virtuelna) adresa?
   Virtuelna adresa ima 48 bitova
b)
c) Veličina straničnog okvira je 4KB
d) U tabelu stranica imamo 1 048 576 ulaza, a u tabelu segmenata 65 536 ulaza
e) Stranični okvir u fizičkoj adresi ima 12 bita
   34-12=22 bit str okvir

1KB= 2^{10}
2KB= 2^{11}
4KB= 2^{12}
8KB= 2^{13}
16KB= 2^{14}
32KB= 2^{15}
64KB= 2^{16}
128KB= 2^{17}
256KB= 2^{18}
512KB= 2^{19}
1024KB= 1MB= 2^{20}

6.2.2015
6.
Fizička memorija = 2^{32} B
Veličina stranica = 2^{10} B
Logički adresni prostor iznosi 2^{16} stranica

25	9	0
Broj stranica	Offset	
64KB	1KB	

Veličina logičke adrese iznosi 26bita
b) u straničnom okviru se nalazi 1024 bajta ili 1kb
c) stranični okvir u fizičkoj adresi iznosi 10 bita 32-10=22
d) u tabelu stranica ima 65 536 ulaza (64k)
e) svaki ulaz u 22b i 3b pomoćno

2MB= 2^{21}
4MB= 2^{22}
8MB= 2^{23}
16MB= 2^{24}

23.4.2014
6.
TLB je keš memorija tabele stranica. TLB se puni istim podacima kao i tabela stranica, s tim što je TLB manjeg kapaciteta i jako brži. Program najpre pristupa TLB-u i proverava da li je odgovarajući stranični okvir adresirane virtuelne adrese u njemu, ako jeste pristupa RAM memoriji kako bi pristupio odgovarajućem podatku, ako ne, mora da pristupi tabeli stranica. Glavni cilj uvođenja TLB-a je ubrzanje pretraživanja tabele stranica, jer svaki memorijski pristup zahteva pristupanje tabeli stranica. Zapravo prevođenje logičke stranice u fizičku bez pristupanja tabeli stranica, koliko je to moguće.

05.2014
6.
Koji su razlozi za uvođenje invertovanih tabela stranica kod 64-bitnih računarske arhitekture u odgovarajući OS-a

Veličina stranica = 2 KB

a) Za ofset u okviru adrese neophodno je 11 bita
b) Tabela stranica ima 16 777 216 ulaza, 2^{24}, a najveći kapacitet virtuelne memorije je 2^{35}=32GB
c) ?
d)

35	11	0

Broj logičkih stranica	Veličina stranice (pomeraj u okviru stranice)
16MB	2KB

02.2017

7. Nakon što OS učita referenciranu stranicu sa diska,mora da je smesti u odgovarajući slobodni stranični okvir u memoriji. Kada se stranica smesti, potrebno je ažurirati tabelu stranica, tako da za datu stranicu ne ukazuje da je na disku, već u memoriji. Adresa straničnog okvira se menja i bit referenciranja.

Brzina učitavanja stranice sa diska = 24ms
Brzina za prekidanje i rest. procesa = 5ms
Brzina izvršenja algoritma za zamenu str = 1ms
Brzina pristupa memoriji = 50ns

$P=10^4$
EFP = ?

EFP= (1-P)*(TLB-HIT)+P*(TLB-MISS)= (1-P)*50ns+P*(24ms+5ms+1ms+50ns)=50ns + P*30ms = =P*30ms +50ns

$EFP=(1-10^{-4})(50ns+50ns+50ns)+10^{-4}*(50ns+50ns+24ms+5ms+1ms+50ms)$
Ne nastaje greška

06.02.2015

7.

Kog algoritma zamene stranica časovnik sa dve kazaljke postoje dva pokazivača (prednja i zadnja kazaljka), a kriterijum za izbor stranica koja će biti zamenjena se zasniva na bitu upotrebe (referenciranja). Kod algoritma časovnik sa jednom kazaljkom kriterijum za izbor stranice koja će biti zamenjena se zasniva na bitu upotrebe i bitu primene. Ovaj algoritam ima jedan pokazivač (kazaljku) koji ukazuje na straničnim okvir od koga se započinje pretraživanje

Stranični okvir	0	1	2	3	4	5	6	7
Stranica	12	10	4	5	7 (14)	13	2	20
Bit upotrebe	1	0	0	1	0	1	1	0

10,14,15,2,18,14,2

1. Stranica 10 je već u memoriji ; ne dolazi do greške stranice, bit upotrebe str okvira 1 postaje 1
2. Stranica 7 se menja stranicom 14, jer je u=0, i sada je u=1
3. Stranica 20 se menja stranicom 15, jer je u=0 i sada je u=1
4. Stranica 2 je već u mem. i ne dolazi do greške stranice
5. Stranica 10 se menja stranicom 18, jer je u=0 i sada je u=1
6. Stranica 14 je već u mem i ne dolazi do greške stranice

7. –||– 2 –||–

05.2014

7. Kod algoritma LRU prilikom greške stranice bira se stranica koja najduže vreme nije bila referencirana, a kod časovnika sa dve kazaljke kandidat za zamenu se određuje na osnovu bita referenciranja (use bit). Efikasnost algoritma zamene stranica se meri po broju grešaka stranica do kojih dolazi primenom algoritma.

3214512642762332173462541 3 ukupno 12 pf

a)LRU

3	3	3	3	5	5	4	4*	3	3	3	3	3	3*	5
	2	2	2	2	2	2	2	2	2	2*	4	4	4	4
		1	1	1	1	1*	7	7*	1	1	1*	6	6	6
			4	4*	6	6	6	6	6	7	7	7*	2	2

b) clock

3
2
1
4

c)FIFO

27.06.2014

7.rešen

9.02.2017

8.

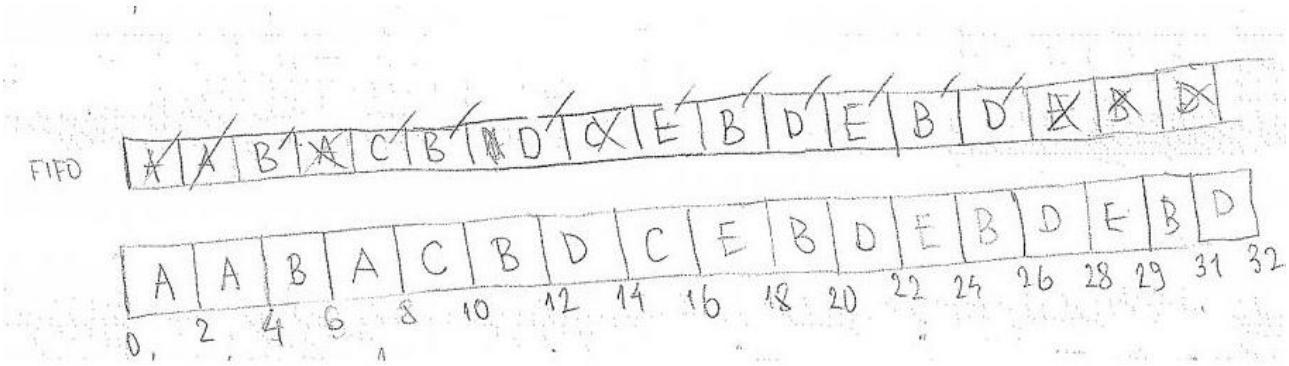
a) p1- jer je U/I orijentisan

b)

Process	BT	AT	TT RR	CT RR	TT SRT	CT SRT
A	~~6~~ ~~4~~ 8 0	0	8	8	6	6
B	~~10~~ ~~8~~ ~~6~~ ~~4~~ 2 0	3	28	31	29	32
C	~~4~~ 2 0	6	10	16	4	10
D	~~7~~ ~~5~~ ~~3~~ 1	9	23	32	18	17
E	~~5~~ ~~3~~ 1 0	12	17	29	10	22

TTavg=17,2

RR g=2



Proces se izvršava manje vremena nego dodeljeni vremenski kvant samo ako je BT<Tg

2.SRT- nakon svakog izvršenja testira se koji od pristiglih procesa ima najmanji BT. Kada svi procesi pristignu biramo

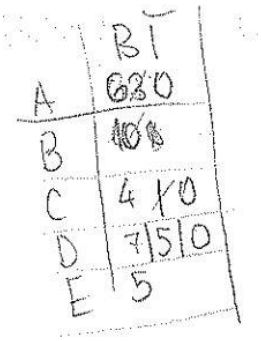
A	A	C	C	D	D	E	B

0 3 6 9 10 12 17 22 32

onog sa najmanjim BT i izvršavamo ga do kraja (SPN)

3. ML feedback g=2 TTavg =13,4

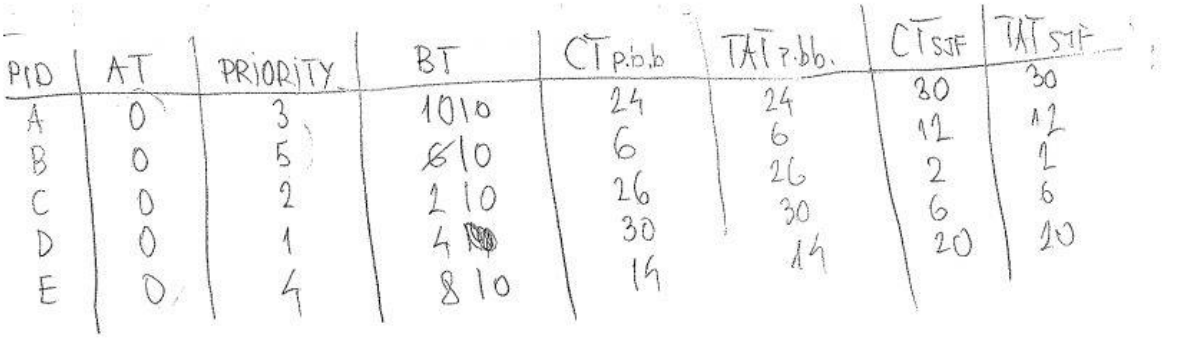
4reda



23.4.2014

8.

SRT algoritam (shortest remaining time)



a) TQ=?

b) Prioritet bez prekidanja

B	E	A	C	b	
0	6	14	24	26	30

TATpbpavg=20

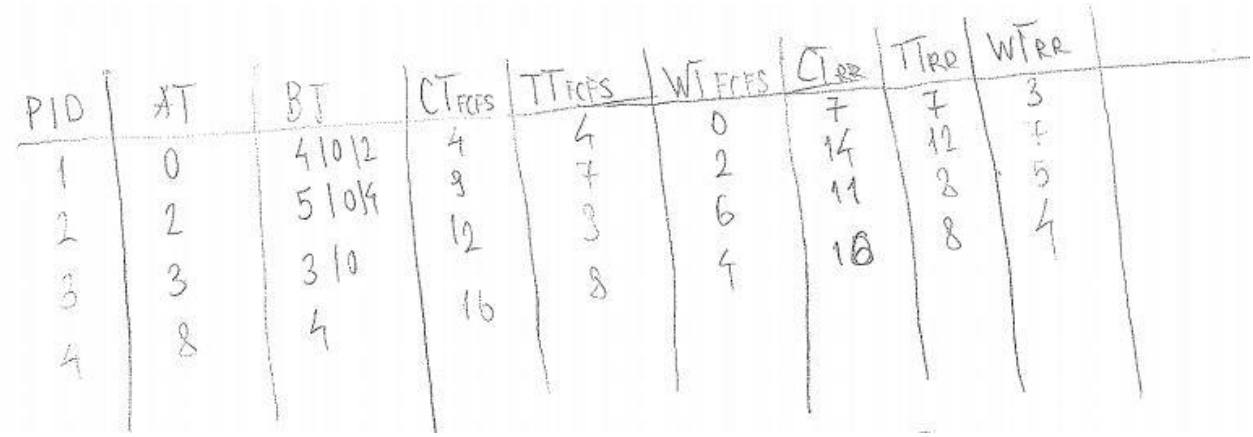
c) SJF

C	D	B	E	A	
0	2	6	12	20	30

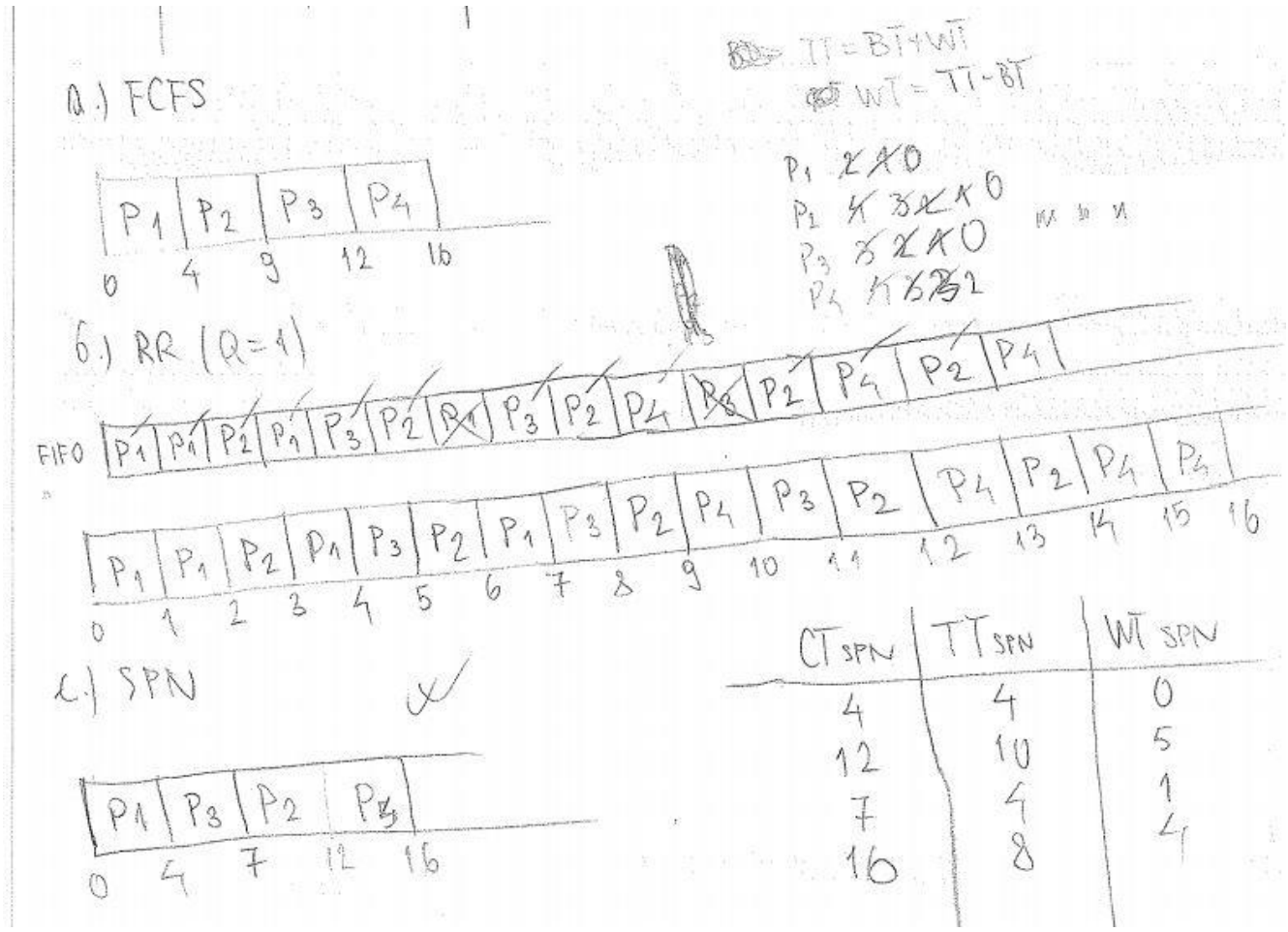
TATsjfavg=20

05.2014.

8. Cilj algoritama SPN i SRT je dati prednost kraćim procesima pri planiranju, tj procesima orjentisanim na U/I. Teški su za implementaciju jer se sa svakim planiranjem troši vreme za proračunavanje najkraćeg procesa



PID	AT	BT	CT_{FCFS}	TT_{FCFS}	WT_{FCFS}	CT_{RR}	TT_{RR}	WT_{RR}
1	0	4|0|2	4	4	0	7	7	3
2	2	5|0|4	9	7	2	14	12	5
3	3	3|0	12	9	6	11	8	5
4	8	4	16	8	4	16	8	4



9.02.2017

9.

a)Sloj U/I softverska fizička organizacija vrši prevođenje logičke adrese zadate u obliku br bloka u fizičku adresu cilindar staza/sektor, a sloj raspoređivanje i upravljanje se vrši kada kontroler diska prekidom označi kraj transfera.

b) zahtevi : 97,123,110,186,147,35,10,84,120 130,35,80
Glava se nalazi na stazi 100
SSF: 97,84,35,35,10,80,84,110,120,129,130,147,186
SCAN:97,84,35,35,10,80,84,10,120,123,130,147,186
F-scan : 97,84,35,10,84,110,120,129,147,186,130,80,35

02.2015
9. a) Da bismo što je više moguće smanjili srednje vreme pozicioniranja glave diska (vreme traženja) i na taj način poboljšali performanse
b) zahtevi: 26,37,100,14,88,33,99,64,68,12 glava se nalazi na 55
12,14,26,33,37,64,67,88,99,100

SSF:64,67,88,99,100,37,33,26,14,12

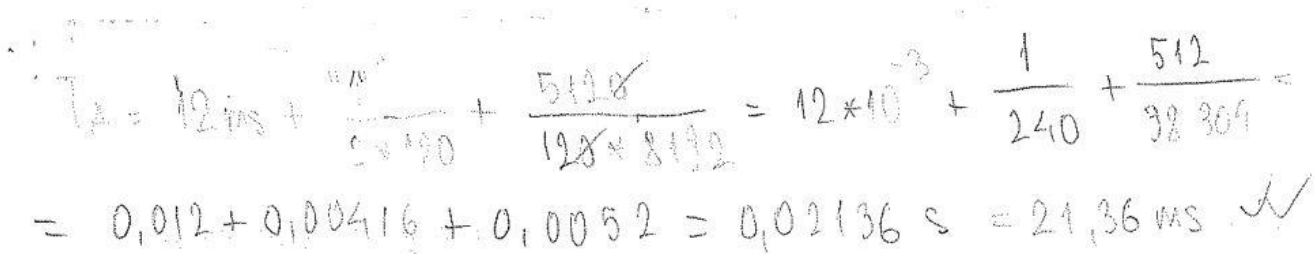
SCAN:64,67,88,99,100,37,33,26,14,12
F-SCAN: 64,67,88,99,100,12,14,26,33,37
F- stepScan(n=3)
26,37,100 47,88,33 99,64,67 12
Stepscan: 100,37,26,33,88,14,64,67,99,12

27.06.2014
9.
Drajver diska je softver koji vrši transformaciju logičkih U/I operacija visokog nivoa u sekvencu U/I instrukcija niskog nivoa razumljivih za kontroler diska. Pored toga drajver diska obezbeđuje formiranje reda čekanja logičkih komandi, prevodi log. vr. bloka u fizičku adresu na disku, vrši obradu greške i komunicira sa kontrolerom diska preko sistemske magistrale. Za razliku od drajvera, kontroler diska je hardverska komponenta i ono prihvata komande drajvera (drajver upisuje komandu u registar kontrolera), obrađuje ih, nakon čega se drajver blokira i čeka prekid od U/I uređaja.

16 sektora po stazi : 512B po sektoru.
Ta=Ts+1/2+b/v*N

V= 7200
Ts= 12ms
Datoteka = 10 blokova
1 blok zauzima 1 sektor
Datoteka = 10*512B=5KB=5120B

a) Kontrolna alokacija



23.04.2014
9.
Staza= 8 sektora
Sektor = 512B
V= 3000 rpm
Ts=15ms
Datoteka=8blokova=8*512=4096B=4KB
1blok=1 sektor

6.1 $T_A = 15ms + \frac{1}{2\cdot 50} + \frac{4096}{50 * 4096} = 15\cdot 10^{-3} + \frac{1}{100} + \frac{1}{50} = 15\cdot 10^{-3} + \frac{3}{100} =$

$= 0{,}015 + 0{,}03 = 0{,}045 s = 45 ms$

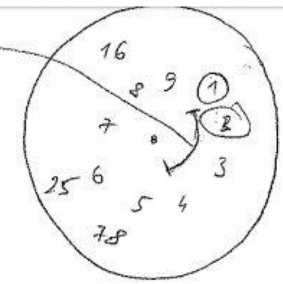
05.2014

9.

9.

0x.05.2014

[1, ②, ⑯, 25, 78, 4].



6.1 индексирање

$T_A = T_s + \frac{1}{2r} + \frac{b}{rN} + 10\cdot\left(T_s + \frac{1}{2r} + \frac{b}{rN}\right)$

$b = 512B$

учитавамо индекс
на основу кога приступамо
осталим блоковима (секторима) на диску

132
45,83
2,86
180,69

$= 11\cdot 12ms + 11\cdot\frac{1}{240} + 11\cdot\frac{512}{120\cdot\frac{8192}{16}} = 132\cdot 10^{-3} + 45{,}83\cdot 10^{-3} + 2{,}86\cdot 10^{-3} =$

$= 180{,}69\ ms$