

Parcijalni ispit 2

I-prva

1. 2018 - OKT

Ukoliko virtualni adresni prostor iznosi 2^{28} stranica od 2KB, koji se mapira u fizičku memoriju od 2^{17} straničnih okvira, koliko: a) bitova iznosi virtuelna adresa, b) bitova iznosi fizička adresa, c) koliko bitova od adrese predstavlja offset, d) koliko stavki (ulaza) ima u invertovanoj tabelli stranica, e) koliko ima pristupa memoriji da bi se pristupilo konkretnom bajtu/reći sa podatkom ukoliko se koristi invertovana tabela stranica i heš funkcija daje jedinstvenu vrednost?

adresni prostor $2^{28} ==$ logicka adresa == virtuelna adresa != fizicka adresa

velicina stranice 2^{11}

broj stranicnih okvira $2^{17} = \text{frame}$

offset 11 bita (velicina stranice)

virtuelni adresni prostor je veci ili jednak od fizickog adresnog prostora

- a) 28 bita - virtuelna adresa je velicine adresnog prostora
- b) 28 bita - fizicka adresa <frame, offset> frame je 17 offset je 11
- c) 11 bita
- d) 2^{17} - logicka adresa <broj stranica, offset>, 28 bita se koristi za logicku adresu, a od toga 11 za offset na osnovu cega se zaključuje da je 17 bita namenjeno za adresiranje stranica (broj stranica i frame nije isto, jer se iz istog okvira moze izvuci stranica i ubaciti nova, pa samim tim broj stranica moze biti veci od broja okvira)
-> broj stavki u tabeli stranica jednak je broju stranica
- e) u knjizi pise 2 tako da 2 - na osnovu hesh funkcije se pristupa invertovanoj tabeli stranica, sto je prvi pristup, i drugi kada na osnovu vrednosti iz tabele stranica se pribavlja stvarni podatak

2. 2018 - SEP

a) Zašto je veličina stranice uvek veličine stepena 2 ?

(b) Navesti razlike između logičke i fizičke adrese.

a) Recall that paging is implemented by breaking up an address into a page and offset number. It is most efficient to break the address into X page bits and Y offset bits, rather than perform arithmetic on the address to calculate the page number and offset. Because each bit position represents a power of 2, splitting an address between bits results in a page size that is a power of 2. Подсјетимо да се страницко позивање проводи раздвајањем адресе на страницу и бројем помака. Најефикасније је разбити адресу на K битове странице и I оффсет битове, уместо да изведете аритметику на адреси за израчунавање броја странице и оффсет. Будући да свака позиција бита представља снагу 2, подјела адресе између битова резултира величином странице која износи снагу 2.

b) The logical address is used like a reference, to access the physical address. The fundamental difference between logical and physical address is that logical address is generated by CPU during a program execution whereas, the physical address refers to a location in the memory unit.

Logička adresa koristi se kao referenca za pristup fizičkoj adresi. Temeljna razlika između logičke i fizičke adrese je ta što logičku adresu generiše CPU tokom izvođenja programa, dok se fizička adresa odnosi na lokaciju u memorijskoj jedinici.

Ukoliko virtuelni adresni prostor iznosi 2^{32} stranica od 4KB, koji se mapira u fizičku memoriju od 2^{20} straničnih okvira, koliko: a) bitova iznosi virtuelna adresa, b) bitova iznosi fizička adresa, c) koliko bitova od adrese predstavlja offset, d) koliko stavki (ulaza) ima u invertovanu tabelu stranica, e) koliko ima pristupa memoriji da bi se pristupilo konkretnom bajtu/reči sa podatkom?

3. 2018 - JAN

broj okvira 2^{20}

virtualni adresni prostor 2^{32}

Stranica 4kb je 2^{12}

- a) $20b+12b=32b$
- b) $20b+12b=32b$
- c) 12b
- d) 2^{20}
- e) 2

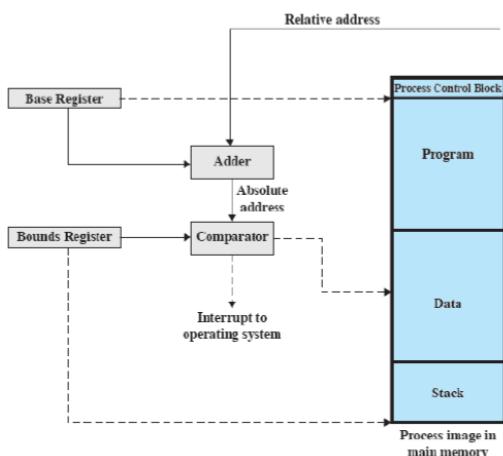
4. 2017- OKT

Odgovoriti i objasniti ukratko:

- a. Šta je relocacija procesa i kako se obavlja u okviru OS?
- b. Koji su razlozi (navesti barem 2) da se dozvoli da dva ili više procesa dele određenu memoriju?
- c. Koja je razlika između interne i eksterne fragmentacije?

a) Relokacija procesa i kako se obavlja u okviru OS?

Relokacija je smeštanje i pomeranje procesa u glavnoj memoriji. Lokacije koje proces referencira nisu fiksne, one će se pomerati svaki put kada se proces zameni, ili pomeri.



U osnovni registar se učita adresa u fizičkoj memoriji kada se proces učita ili zameni. Sve vreme izvršenja programa radi se sa relativnim adresama. U toku izvršenja se koristi bazni i granični registar. (316. str. u knjizi)

b) Ako više procesa izvršava isti program, pogodno je da svaki program pristupa istoj kopiji programa, a ne da ima sopstvenu kopiju.

Procesi koji sarađuju na istom zadatku treba da pristupaju zajedničkoj strukturi podataka.

c) Interna fragmentacija se javlja kod particija fiksnih veličina. Kada se program učita u memoriju najverovatnije neće ispuniti celu particiju alii će se ta particija voditi kao zauzeta. Taj neiskorišćeni, a navodno zauzeti deo, se naziva interna fragmentacija.

Eksterna fragmentacija se javlja kod dinamičkog deljenja particija. Kod eksterne fragmentacije ukupna velicina slobodnih particija je dovoljna za smestanje procesa, ali je nekontinualna u memorijskom prostoru.

5. 2017- JAN

U operativnom sistemu je implementirano upravljanje memorijom korišćenjem segmentacije sa straničenjem i jednom tabelom stranica po segmentu. Ukoliko je fizička memorija veličine 2^{34} B, maksimalni broj segmenata 64K, maksimalna veličina segmenta je 2^{20} stranica, a stranice su veličine 2^{12} B:

- a) Koliko bitova ima logička (virtuelna adresa)?
- b) Koliko je format logičke (virtuelne) adrese? Obrazložiti.
- c) Koja je veličina straničnog okvira?
- d) Koliko ima ulaza u tabelu segmenata, a koliko u tabele stranica?
- e) Koliko bitova u fizičkoj adresi specificira stranični okvir?

fizička memorija – 2^{34} B,
broj segmenata - $64k = 2^{16}$
veličina segmenata - 2^{20} stranica
Stranice veličine - 2^{12} B

Čim imaš segmentaciju sa straničenjem zamisli je kao tablicu stranica u 2 nivoa.

- a) **Virtuelna adresa: $16b + 20b + 12b = 48b$**
broj segmenta*velicina segmenta (isto sto ibroj stranica)*velicina stranice

- b) **Bolje je pitanje Kakav je format:**

16b	20b	12b
Segment	Broj stranica	Veličina stranica

- c) **Veličina straničnog okvira je jednaka veličini stranice => $2^{12} = 4KB$**

d) 2^{16} ulaza u tabelu segmenata, jer ima 64k segmenata

A u tabelu stranica 2^{20} jer je max veličina segmenta 2^{20} stranica.

Tabela segmenata ima 2^{16} ulaza. Svaki od tih ulaza ima po 20b, sto znaci da ima 2^{20} stranica. Na tako očitanu vrednost iz tabele segmenata konkateniraš offset iz virtuelne adrese i dobio si fizičku adresu.

e) Ukupna fizička memorija je adresirana sa 34b, od toga je 12b otpalo na adresiranje unutar stranice, odakle sledi da je 22b zaduženo za adresiranje stranice, odnosno straničnog okvira.

6. 2016 - OKT2

Koja je glavna razlika u upravljanju memorijom između segmentacije i segmentacije pomoću straničenja i na koji način se obavlja prevodenje logičke u fizičku adresu u ove dve metode. Ako za određeni proces imamo sledeću tablicu segmenata, za svaku od navedenih logičkih adresa odrediti njenu fizičku adresu ili naznačiti ukoliko je nastala greška segmenta.

Segment	Baza (početna adresa)	Limit (veličina)	a) 0, 430
0	1219	600	b) 1, 15
1	3300	14	c) 2, 50
2	90	100	d) 3, 400
3	2327	580	e) 4, 112
4	1952	96	

Indici kraljevi znaci ovako

- a) za 0-ti segment u tabeli uzmes limit i gledas da li je offset<limit, $430 < 600$ znaci da nema greske. Fizicka adresa je offset+base $430 + 1219 = 1649$, tako za sve.

Za segmentaciju: iz logicke adrese se uzima deo za segment i pristupa se tabeli segmenta odkle se uzima base i sabira sa ofsetom. Na taj nacin se obavlja prevodjenje u fizicku.

Za segmentaciju sa stranicenjem adresa je oblika <segment, stranica, offset>. Pomocu segmenta se pristupa tabeli segmenata i uzma base koji se dodaje stranici i na osnovu dobijene vrednosti se pristupa odgovarajucoj adresi u tabeli stranica gde se nalazi frame (f). Fizicka adresa je oblika <frame, offset>.

7. 2016 - OKT

Prepostavite da imate virtuelne adrese od 47bitova za adresiranje virtuelnog adresnog prostora sa veličinom stranice od 16 K, pri čemu svaki ulaz u stranicu je veličine 8 bajtova. Koliko nivoa tablica stranica je neophodno da bi se mapirao virtuelni adresni prostor, ako se zahteva da svaka tabela stranica bude smeštena u jednu stranicu. Objasniti detaljno i dati strukturu virtuelne adrese. Ako je veličina fizičke memorije 1GB koliko bitova je potrebno za svaki ulaz u tablicu stranica?

<page, offset>

<page1,page2,page3,page4...,offset>

- a) Koliko nivoa tablica stranica je potrebno da bi se mapirao v.a. prostor ako se zahteva da svaka tabela stranica bude smeštena u 1 stranicu?

14b je za offset. Svaki ulaz u tabelu stranica je 8B, što znači da jedna tabela stranica ima 2^{11} ulaza/stavki. Jer je veličina tablice jednaka veličini stranice odnosno 2^{14} , $2^{14} / 2^3 = 2^{11}$. Znači treba nam 11 bitova u adresi da indeksiramo jednu tabelu stranica ako ima 2^{11} stavki. 11b za jednu tabelu stranica, 11b za drugu, 11b za direktorijum.

$$11b + 11 + 11 + 14 = 47$$

Znači 3 nivoa je odgovor.

b) Objasniti detaljno i dati strukturu virtuelne adrese.

Virtuelna adresa je u formatu <stranica, pomeraj>. Stranica se koristi pri pristupanju Tabeli stranica na osnovu koje se određuje fizička adresa odnosno stranični okvir kome se pristupa sa pomerajem navedenim u drugom delu adrese. Ovde se za stranicu koristi 33b, a za pomeraj 14b. Ipak, kako se koristi adresiranje u 3 nivoa, prva 33b biće podeljeni na 3 dela, po jedan za pristup svakom nivou.

c) Veličina fizičke memorije je 1GB, koliko b je potrebno za svaki ulaz u tablicu stranica?

$$1\text{GB} = 2^{30}$$

$30 - 14 = 16b \Rightarrow 2^{16}$ straničnih okvira je u toj fizičkoj memoriji, pod pretpostavkom da i dalje važi da je veličina stranice 16KB.

Tako da je moj konačan odgovor da je potrebno 16 + ona 3 b, kako bi se adresirali svi stranični okviri.

8. 2016 - SEP

U operativnom sistemu je implementirano upravljanje memorijom korišćenjem straničenja u dva nivoa. Ukoliko je fizička memorija veličine 2^{34} B, stranice su veličine 2^{16} B, a ulaz u tablicu stranica 4 B odrediti:

- a) Koliko bitova ima logička (virtuelna) adresa?
- b) Koliko bitova iznosi offset?
- c) Koja je veličina straničnog okvira?
- d) Koliko ima ulaza u tablicu stranica prve i druge nivoa?
- e) Koliko bitova u fizičkoj adresi specificira stranični okvir?

fizicka memorija 2^{34}

velicina stranice 2^{16}

ulaz u tabelu stranica 2^2

Nemam pojma da li je ovo dobro

<p1,p2,offset>

- a) 44b
 - b) 16b
 - c) 2^{16} ? okvir je za obicnu tabelu jednak velicini stranice
 - d) ukupno 2^{28} prva 2^{14} druga 2^{14}
 - e) 18
-

9. 2016 - JUN

Pretpostavite da treba da projektujete sistem virtuelne memorije sa sledecim karakteristikama

- a. Veličina ulaza u tabelu stranica je 4 B
- b. Svaka tabela stranica mora da se uklopi u jedan stranični okvir
- c. Sistem mora da podrži virtualni adresni prostor od 2^{32} B (256 GB)

Pretpostavite da želite da koristite straničenje u više nivoa sa ne više od 2 nivoa. Koja je minimalna veličina stranice u tom slučaju? Objasniti.

Nacrtati strukturu virtuelne adrese po poljima koja treba da koristi MMU sistem za prevođenje adresa. Ukoliko implementirate straničenje u 3 nivoa, koja je minimalna veličina stranice? Objasniti! Nacrtati strukturu virtuelne adrese u ovom slučaju!

10. 2016 - APR

1. Sistem implementatora virtuelnu memoriju korišćenjem straničenja sa tablicama stranica u jednom nivou. Maksimalna veličina adresnog prostora je 16MB. Tablica stranica procesa koji se izvršava je:

stranica	Stranični okvir
0	4
1	8
2	16
3	17
4	9

Veličina stranice je 1024B, maksimalna veličina fizičke memorije je 2MB. Koliko bitova je potrebno za svaki ulaz u tablicu stranica? Koliki je maksimalan broj ulaza u tablici stranica? Koliko bitova ima virtuelna adresa? U koju fizičku adresu se preslikava virtuelna adresa 1524. Koja virtuelna adresa se preslikava u fizičku adresu 10020?

$16\text{Mb} = 2^{24} \rightarrow 24$ bita logicka adresa

offset je jednak velicini stranice = 10b

znaci ostajih 14 za adresiranje broj ulaza je 2^{14}

Broj straničnih okvira je fizicka memorija/offset = $2^{21} - 10 = 11$, odnosno 2^{11} straničnih okvira.

An address space is a range of valid addresses in memory that are available for a program or process. That is, it is the memory that a program or process can access. The memory can be either physical or virtual and is used for executing instructions and storing data.

Knjiga: Adresni prostor je opseg adresa dostupnih računarskom programu.

Koliko b je potrebno za svaki ulaz u tablicu stranica?

Treba nam 11b za svaki stranični okvir + 3b za present/absent, bit referenciranja i bit modifikacije.

Koliko b ima virtuelna adresa?

24b jer je adresni prostor 16MB, a to je 2^{24} , ako se adresira svaki bajt. 11b je stranicni okvir, 10b je offset, 3b gorenavedena. Adresni prostor je memorijski prostor koji mi hocemo da adresiramo u tom procesu.

U koju fizičku adresu se preslikava virtuelna adresa 1524?

Tipična greška je prevođenje u binarni.

1524 podeliti sa 1024, a to je 500-ti bajt stranice 1. Stranica broj 1, offset je 500. Gledamo sad tabelu, stranica 1 je slikana u stranični okvir broj 8. $8 \times 1024 = 8192 + 500 = 8692$ to je fizička adresa.

Koja virtuelna adresa se preslikava u fizičku 10020?

10020 podelimo sa 1024 i dobijemo da je to stranični okvir 9 i offset 804. Stranični okvir 9 je virtuelna stranica 4. Virtuelna adresa je $4 \times 1024 + 804 = 4900$.

11. 2016 - MAR

Prepostavite da imate virtuelne adrese od 47bitova za adresiranje virtuelnog adresnog prostora sa veličinom stranice od 16 K, pri čemu svaki ulaz u stranicu je veličine 8 bajtova. Koliko nivoa tablica stranica je neophodno da bi se mapirao virtuelni adresni prostor, ako se zahteva da svaka tabela stranica bude smeštena u jednu stranicu. Objasnit detaljno i dati strukturu virtuelne adrese. Ako je veličina fizičke memorije 1GB koliko bitova je potrebno za svaki ulaz u tablicu stranica?

uradjen gore

12. 2016 - JAN

- a) Koja je glavna prednost koštanja tabele stranica u više nivoa u odnosu na jedinstvenu tabelu stranica?
- b) Prepostaviti da računarski sistem ima 38-bitne virtuelne adrese i 22-bitne fizičke adrese. Na tabelama stranica u dva nivoa, na stranicama veličine 16KB i ulazima u tabele stranica od 4B, nevini format virtuelne adrese.
- c) Ako sistem ima stranice od 8KB, 256 KB glavne memorije i virtualni adresni prostor od 4GB i koristi invertovano tabelu stranica, kolika treba da bude veličina hash tabele da bi srednja duljina hash lanca bila manja od 1 (veličina hash tabele je 2^n)?

13. 2015 - APR

Sistem implementatora virtuelna memoriju koristenjem stranicenja sa tablicama stranica u jednom nivou. Maksimalna veličina adresnog prostora je 16MB. Tablica stranica procesa koji se izvršava je:

Stranica	Stranicni okvir
0	4
1	8
2	16
3	17
4	9

Velicina stranice je 1024B, maksimalna velicina fizičke memorije je 2MB. Koliko bitova je potrebno za svaki ulaz u tablicu stranica? Koliki je maksimalan broj ulaza u tablici stranica? Koliko bitova ima virtuelna adresa? U koju fizičku adresu se preslikava virtuelna adresa 1524. Koja virtuelna adresa se preslikava u fizičku adresu 10020?

uradjen gore

14. 2015 - MAR

Prepostavite da imate virtuelne adrese od 47bitova za adresiranje virtuelnog adresnog prostora sa veličinom stranice od 16 K, pri čemu svaki ulaz u stranicu je veličine 8 bajtova. Koliko nivoa tablica stranica je neophodno da bi se mapirao virtuelni adresni prostor, ako se zahteva da svaka tabela stranica bude smeštena u jednu stranicu. Objasnit detaljno i dati strukturu virtuelne adrese. Ako je veličina fizičke memorije 1GB koliko bitova je potrebno za svaki ulaz u tablicu stranica?

uradjen

15. 2015 - JAN

Razmotriti jednostavan sistem za upravljanje memorijom korišćenjem straničenja sa sledećim parametrima: 2^{32} B fizičke memorije, veličina stranice 2^{10} B, logički adresni prostor iznosi 2^{16} stranica.

- a. Koliko bitova je veličina logičke adrese?
- b. Koliko bajtova se nalazi u straničnom okviru?
- c. Koliko bitova u fizičkoj adresi specificiraju stranični okvir?
- d. Koliko ima ulaza u tabelu stranica?
- e. Koliko bitova se nalazi u svakom ulazu tabele stranica?

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

$$2\text{MB} = 2^{21}$$

$$4\text{MB} = 2^{22}$$

$$8\text{MB} = 2^{23}$$

$$16\text{MB} = 2^{24}$$

II-druga

1. 2018 - OKT

(a) Koja je sličnost između FIFO i Clock algoritama za zamenu stranica?

(b) Ukoliko je pokazivač clock-a na okviru 0, a „najstarija“ stranica u memoriji je 14, koje bi stranice bile zamenjene ovim algoritmima i kako bi izgledala tabela posle zamene.

Stranični okvir	0	1	2	3	4	5	6	7
Stranica	14	36	11	5	7	13	2	20
Bit upotrebe	1	1	1	1	0	1	1	0

- a) Ta da je clock algoritam jednostavna modifikacija FIFO algoritma koja pored starosti stranica ima i u- bit koji se postavlja na 1 kad god je stranica ucitana ili referencirana

-Ovo vam je vise neka razilika, ovako nesto sam ja nasao(iz knjige):

FIFO tretira okvire stranica dodeljenje procesu kao kruzni bafer. Pointer kruzi kroz okvire procesa, zamenjuje onaj koji je najduze u memoriji

CLOCK posmatra skup svih okvira koji su kandidati za zamenu kao kruzni bafer kome je pridruzen pointer(pointer pokazuje na sledeci okvir posle azuriranog). Clock postaje FIFO ako su mu svi use bitovi na 1

Uzimaju "starost" stranica

Znaci razlika im je kako menjaju stranice, ali oba nacina imaju kruzni bafer,fifo menja first in, ovaj prvi na koji naidje sa 0

- b) FIFO bi isao redom prva stranica koja bi bila izbacena je 14 i pokazivac bi presao na 36. Sledeca stranica koja bi bila zamenjena je 36 nakon cega bi pokazivac otisao na 11 i tako do kraja tabele
- Clock algoritam bi prosao kroz tabelu i trazio stranice ciji je bit upotrebe 0 u nasem slucaju prva stranica koja bi bila zamenjena je 7 nakon nje sledeca na redu je 20. U tabeli sada samo 2 nove stranice imaju bit upotrebe 1 a sve ostale su postavljene na 0

2. 2018 - SEP

- a) U kom slucaju nastaje grecka stranice i šta OS radi u tom slucaju?
b) Ukoliko OS implementira LRU strategiju zamene stranice, koje bi stranice bile zamenjene ovim algoritmom ukoliko se dogode tri uzastopne grecke stranice.

Stranicni okvir	0	1	2	3	4	5	6	7
Stranica	14	36	11	5	7	13	2	20
Vreme referenciranja	1010	1011	0101	1100	0011	0110	1000	0111

- a) Ukoliko proces referencira neku stranicu koja se ne nalazi u tabeli stranica, OS pribavlja stranicu sa diska i upisuje je u slobodan stranicni okvir, ako ne postoji slobodan okvir on mora na osnovu nekog algoritma da odabere koju ce stranicu izbrisati i na njeno mesto upisati novu stranicu nakon cega treba azurirati tabelu.
- b)
- | | |
|-----------|----|
| 1010 - 10 | 14 |
| 1011 - 11 | 36 |
| 0101 - 5 | 11 |
| 1100 - 12 | 5 |
| 0011 - 3 | 7 |
| 0110 - 6 | 13 |
| 1000 - 8 | 2 |
| 0111 - 7 | 20 |

7, 11, 13 Ove 3 nisu skoro koriscene pa se prema ovom algoritmu one prve menjaju

3. 2018 - JAN

- a) Da li se i sto menja u tabeli stranica tokom obrade grecke stranice?
b) Zašto je u savremenim OS (npr. Linux, Solaris) implementirana varijacija algoritma Clock, a ne klasičan LRU ili optimalan algoritam?

- a) Tokom obrade greske u tabeli stranica je potrebno upisati novu stranicu koja je referencirana ako u njoj nema slobodnih stranicnih okvira potrebno je jednu od postojećih stranica izbrisati i na njeno mesto ubaciti novu nakon toga azurirati tabelu. Menja se sadrzaj tabele stranica.

b) Optimalni algoritam je nemoguce implementirati (u trenutku greske OS ne moze da zna kada ce koja stranica u memoriji biti referencirana), a LRU bi morao da prodje kroz celu tabelu stranica kako bi pronasao onu koja je najdavnije koriscena. Varijacija algoritma Clock sa 2 kazaljke koja se koristi u ovim sistemima pruza brzu zamenu stranica od LRU algoritma i zato je bolja.

b) Mozda bolji odgovor za LRU je da je on tezak za implementaciju jer pored stranica mora da sacuva i vreme kada je ta stranica poslednji put referencirana i to dovodi do troškova.

4. 2017- OKT

- a) Da li korišćenje TLB smanjuje broj grešaka stranica? Ako da objasniti kako, a ako ne objasniti zašto.
- b) Razmotriti dvodimenzionalno polje int A[][] = new int[100][100], gde se A[0][0] nalazi na adresi 200 u stranicenoj memoriji procesa, veličine stranice 200. Kod procesa se nalazi na adresama 0-199. Ukoliko imamo 3 stranična okvira, koliko se grešaka stranice javlja pri izvršenju sledećeg koda za inicijalizaciju matrice korišćenjem LRU algoritma:
- ```
for (int j = 0; j < 100; j++)
 for (int i = 0; i < 100; i++)
 A[i][j] = 0;
```
  - ```
for (int i = 0; i < 100; i++)
    for (int j = 0; j < 100; j++)
        A[i][j] = 0;
```

Mi imamo u 1. straničnom okviru kod procesa i to ne smemo da menjamo, znači ostaju nam 2 stranice gde možemo da upisujemo matricu.

Imamo 10.000 elemenata, stranica veličina 200, pa su ti elementi su u 50 stranica.

Svake 2 vrste su u jednoj stranici pošto je u C-u po vrstama smeštanje.

a) Za $j = 0$ imamo 50 grešaka stranica, za $j = 1$ ponovo 50 grešaka stranica. Znači 50×100 ukupno 5000 grešaka stranica.

b) Smestimo prvu i drugu stranicu i nemamo greške stranica po petlji jot. Kad je $i = 2$ i 3 smeštamo u drugi stranični okvir i nemamo greške po jot. Kad je $i = 4$ i 5, 6 i 7 biće grešaka... Poenta: Imamo grešku stranica posle svake 2 vrste, što znači da imamo 50 grešaka stranica.

Napomena: Isuviše je komplikovano, verovatno neće biti.

5. 2017 - APR

Za šta služi TLB i da li korišćenje TLB smanjuje broj grešaka stranica i kako? Nacrtati i objasniti prevodenje virtuelne u fizičku adresu u sistemu se upravljanje memorijom korišćenjem tabela stranica u dva nivoa, uz korišćenje TLB. Vreme za pristup glavnoj memoriji je T. TLB-u se pristupa veoma brzo, tako reći trenutno ($T/5$ vremenskih jedinica). Aktivni proces pristupa memoriji po virtuelnim adresama. Koliki mora da bude procenat pogodaka u TLB-u ukoliko želimo da pristup memoriji bude prosečno 1.6 T?

6. 2017- APR (dao je dva pitanja iz druge oblasti)

- 2.) Proces zauzima 5 straničnih okvira u memoriji. Vreme učitavanja stranice u svaki od straničnih okvira, vreme poslednjeg pristupa stranici, R bit referenciranja (korišćenja) i M bit modifikacije su prikazani u tablici.

Broj virtuelne stranice	Stranični okvir	Vreme učitavanja	Vreme referenciranja	R bit	M bit
0	2	6	31	0	1
1	1	13	30	1	1
2	0	3	35	1	0
3	3	2	37	1	0
4	4	9	33	0	1

U trenutku 40 nastaje greška stranice za virtualnu stranicu 5. Koja će stranica procesa biti zamjenjena za svaki od algoritama i objasniti zašto a) FIFO b) LRU c) Clock d) Optimalni

- a) 3, 2, 0, 4, 1 FIFO menja se 3
- b) stranica 1 se menja
- c) * prvi prolazak trazi 0,0
* drugi korak trazi 0,1 i setuje r na 0
* ide iz pocetka
- Menja se stranica 0
- d) Ne moze da se odredi

7. 2017- JAN

Da li se i šta menja u tabeli stranica tokom obrade greške stranice? Ukoliko vreme učitavanja jedne stranice sa diska iznosi 24ms, vreme neophodno za prekidanje i ponovno restartovanje procesa 5ms, vreme izvršenja algoritma za zamenu stranice 1 ms, a vreme pristupa memoriji 50ns, koliko je efektivno (srednje) vreme pristupa memoriji korišćenjem straničenja u dva nivoa ukoliko na svakih 10^4 memorijskih referenci nastane jedna greška stranica. Objasniti.

Pri obradi greške se referencira sekundarna memorija kako bi se pribavila odgovarajuća stranica, koja se zatim smešta u fizičku (RAM) memoriju, a njena fizička adresa se upisuje u Tabelu stranica ukoliko ima mesta; ako mesta nema onda se jednim od algoritama zamene bira stranica na čije mesto će biti upisana novopribavljena stranica.

Vreme učitavanja stranice sa diska je 24ms, prekidanje i ponovno restartovanje procesa je 5ms, vreme izvršenja algoritma za zamenu stranice je 1ms, vreme pristupa memoriji 50ns. Odrediti efektivno vreme, ako se koristi straničenje u 2 nivoa i na svakih 10^4 memorijskih referenci nastaje 1 greška stranica. Objasniti.

$$p = 1/10^4 = 10^{-4} \text{ verovatnoća nastajanja greške}$$

$$\text{EFP(efektivno vreme pristupa)} = (1 - p) * \text{Pogodak} + p * \text{Promašaj} = (1 - 10^{-4}) * (50\text{ns} + 50\text{ns} + 50\text{ns}) + 10^{-4} * (50\text{ns} + 50\text{ns} + 24\text{ms} + 5\text{ms} + 1\text{ms} + 50\text{ns})$$

Pri pretraživanju tabele stranica postoje 2 moguća scenarija:

Prvi, da nije nastala greška ($1 - p$) I (zato puta) tada se pristupa prvom nivou Tablice straničenja, pa drugom, pa memoriji, otud $3 * 50\text{ns}$. Objasnili smo deo: $(1 - p) * \text{Pogodak}$

ILI, zato ide plus, nastala je greška I (otud puta) pristupamo dvama (hehe) nivoima gde na jednom uočavamo grešku stranice, opslužujemo tu grešku i nastavljamo pristup memoriji. Objasnili smo: p*Promašaj

8. 2016 - OKT2

Koja je razlika između korišćenja TLB (asocijativnih registara) i Hash funkcije u ubrzavanju pristupa memoriji korišćenjem straničenja? Objasniti. Ukoliko je tablica stranica u glavnoj memoriji, izračunati i objasniti:

- a. Ako je za pristup memoriji potrebno 200ns, koliko je vremena neophodno za pristup podatku u memoriji u okviru stranice, ako su sve stranice u glavnoj memoriji?
- b. Ako se 75% memorijskih referenci stranica nalazi u TLB, a vreme potrebno za pristup TLB iznosi 10ns, koliko je prosečno vreme pristupa podatku u memoriji?

Da bi se izbegao dodatno referenciranje memorije uz svaki memorijski pristup koristi se TLB. Takođe, TLB vrši paralelno pretrazivanje svih stavki po datom kljucu.

Broj virtualne stranice se preslikava u heš vrednost korišćenjem heš funkcije. Heš vrednost predstavlja pokazivač na invertovanu tabelu stranica čija svaka stavka sadrži lančanu listu elementa formata (virtualne stranica, stranični okvir)

Hardverska komponenta u okviru MMU za transformisanje (prevođenje) virtualnih u fizičke adrese bez pristupa tabeli stranica

Razlika je u tome sto kod TLB-a se prvo pristupa TLB komponenti kako bi se pronasla stranica, ako ona postoji u TLB-u nema dalje potrebe za pristup tabeli stranica. Kod hes funkcije koristimo takozvanu hesiranu tabelu u kojoj na osnovu hes vrednosti pristupamo odgovarajućoj stavci u tabeli, svaka stavka predstavlja lankanu listu tipa <stranica, okvir>.

- a) Potrebno je 400ns, prvi pristup tabeli stranica a drugi pristup memoriji
- b) EFP = $0,75*(10\text{ns}+200)+0,25*(10\text{ns}+400\text{ns})$

9. 2016 - OKT

Šta je to greška stranice, kada nastaje i kako se rešava? Ukoliko vreme učitavanja jedne stranice sa diska iznosi 10ms, vreme neophodno za prekidanje i ponovno restartovanje procesa 1ms, vreme izvršenja algoritma za zamenu stranice 1 ms, a vreme pristupa memoriji 100ns, koliko je efektivno vreme pristupa memoriji korišćenjem straničenja ukoliko na svakih 100000 memorijskih referenci nastane jedna greška stranica. Objasniti.

Greska stranice je izuzetak koji generise hardver kada aktivni proces referencira tj pristupa stranici koja se ne nalazi u memoriji.

$$p=1/10^5$$

$$EFP = (1-p)*Pogodak+p*Promasaj$$

$$pogodak= (100\text{ns}+100\text{ns})$$

$$Promasaj=(100\text{ns}+100\text{ns}+10\text{ms}+1\text{ms}+1\text{ms})$$

10. 2016 - SEP

Navesti osnovne strategije algoritama zamene stranica a) FIFO b) LRU c) Clock d) Optimalni.
Procес razumira 4 stranična okvira u memoriji. Vreme učitavanja stranice u svaki od straničnih okvira, vreme poslednjeg pristupa stranici, R bit referenciranja (koriscenja) i M bit modifikacije su prikazani.

U tabeli:

Vreme učitavanja	Stranično okvir	Vreme učitavanja	Vreme referenciranja	R bit	M bit
2	0	60	160	1	1
1	1	130	161	0	0
0	2	26	162	1	0
3	3	20	165	0	1
4	4	85	168	1	1

U trenutku 174 nastaje greška stranice za virtuelnu stranicu 5. Koja će stranica procesa biti zamjenjena za svaki od algoritama i objasniti zašto a) FIFO b) LRU c) Clock d) Optimalni.

- a) 3 (vreme ref-vreme ucitavanja)
- b) 2 (vreme referenciranja najmanje)
- c) 1 (gledaju se nule za bit)
- d) ne moze

11. 2016 - JUN

Za šta služi druga kazaljka u algoritmu Clock sa dve kazaljke? Preposlaviti da sistem ima 3 stranična okvira i da se stranice procesa referenciraju u sledećem nizu:

0 1 2 4 5 1 2 4 | 1 3 5 0 3 1 4 2

Prikazati sadžaj memorije (okvira) i naznačiti greške stranica, kao i ukupan broj gresaka stranica ukoliko se primenjuje strategija a) LRU, b) Clock c) optimalni d) FIFO

Druga kazaljka prolazi kroz buffer i trazi stranicu sa bitom upotrebe 0 kako bi je zamenila.

NE IDE 0,1,2,3,4 NEGO IDE BEZ 3 !!!!!!! NICE!

LRU - 13 gresak(kad se ubaci opet restartuje se brojac za njega, levo)

FIFO - 13 gresaka (najvise koriscen,desno radjen)

A| 0 1 2 3 4 1 2 4 1 2 5 0 3 1 4 2 B| 0 1 2 3 4 1 2 4 1 2 5 0 3 1 4 2

A| 0 0 0 3 3 3 2 2 2 2 2 3 3 3 2

A| 0 0 0 3 3 3 2 2 2 2 2 3 3 3 2

B| 1 1 1 4 4 4 4 4 4 5 5 5 1 1 1

B| 1 1 1 4 4 4 4 4 4 5 5 5 1 1 1

C| 2 2 2 1 1 1 1 1 1 0 0 0 4 4

C| 2 2 2 1 1 1 1 1 1 0 0 0 4 4

Clock (zvezdica u=1) 13 gresaka

0 1 2 3 4 1 2 4 1 2 5 0 3 1 4 2

A| 0* 0* 0* 3* 3* 3* 2* 2* 2* 2* 2 2 3* 3 3 2*

B| 1* 1* 1 4* 4* 4 4* 4* ->4* 5* 5* 5* 1* 1* 1*

C| 2* 2 2 1* 1 1 1* 1* 1 0* 0* 0 4* 4*

Optimalni 9 gresaka (koji ti je najdavnije potreban)

0 1 2 3 4 1 2 4 1 2 5 0 3 1 4 2

A| 0 0 0 3 4 4 4 4 4 4 4 4 4 4 4

B| 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

C| 2 2 2 2 2 2 2 5 0 3 3 3 2

Proveri za poslednji gde ces

2. Prepostaviti da se stranice u virtuelnom adresnom prostoru referenciraju sledećim redom:

1 2 1 3 2 1 4 3 1 1 2 4 1 5 6 2 1

Postoje tri raspoloživa stranična okvira i implementirano je straničenje na zahtev. Prikazati sadržaj memorije nakon svake memoriske reference prepostavljajući da se koristi a) LRU algoritam b) Clock algoritam, c) Optimalni, i prikazati koliko se grešaka stranica obavlja u svakom od ovih slučajeva.

a) LRU

1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	2	2
	2	2	2	2	2	2	2	3	3	3	3	4	4	4	4	6	6
			3	3	3	4	4	4	4	2	2	2	2	5	5	5	1

F F F

Ukupno 11 grešaka kod LRU.

Objašnjenje: Gledaš na levo, šta je najdavnije referencirano i to menjaš.

b) Clock

Polazimo od toga da se ne menja pokazivač pri referenciranju stranice koja je unutar bilo kog od 3 straničnog okvira. Znači, ako je pokazivač na prvoj vrsti a traži se stranica u trećoj ta stranica će moći da bude preuzeta, a da pokazivač ostane gde jeste. Pogledaj prezentaciju 'Virtuelna memorija', slajd 44. poslednju kolonu za Clock. Tu se to vidi.

Napomena: Obavezno crtaj strelicu gde ti se nalazi pokazivač!

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

F F

F

F

F

F

F

2 da napises nema ovakav primer! (zvezdice kad se napuni bafer se restartuju)

12132143112415621

Devet grešaka kod Clock-a.

Objašnjenje: Imaš pokazivač koji se šeta pri traženju stranice. Vidi gorepomentu sliku.

I nakon referenciranje i učitavanja stranice pokazivač se pomera na narednu poziciju.

Jako bitno!

c) Optimalni

Redosled referenciranja: 1 2 1 3 2 1 4 3 1 1 2 4 1 5 6 2 1

1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	2	2	2	2	2	4	4	4	4	4	4	4	5	6	6	6	6
			3	3	3	3	3	3	3	2	2	2	2	2	2	2	2

F

Sedam grešaka u optimalnom algoritmu.

Objašnjenje: Gledaš šta je najkasnije referencirano od 3 stranice koja imaš u straničnim okvirima. Gledaš na desno.

13. 2016 - MAR

Koja je razlika između planiranja bez prekidanja (*non-preemptive*) i planiranja sa prekidanjem (*preemptive*) i kada se obavlja prekidanje? Razmotriti sistem za 3 stanična okvira i sledećom sekvencom

5 6 9 3 5 6 3 6 9 4 3 9 6 4 9

Koliko grešaka stranica bi bilo generisano ako bi se koristio a) FIFO, b) LRU i c) optimalni algoritam zamene stranica. Da li bi se smanjio broj grešaka stranica ako bi se koristio TLB i na koji način?

– Bez prekidanja (nonpreemptive) • Kada je proces u stanju izvršenja, izvršava se sve dok se ne terminira (inicijalizuje) ili dok se ne blokira zbog UI

– Sa prekidanjem (preemptive) • OS može prekinuti proces koji se izvršava i premestiti ga u red spremnih procesa • Omogućava bolje usluge jer nijedan proces ne može dugo monopolizovati procesor

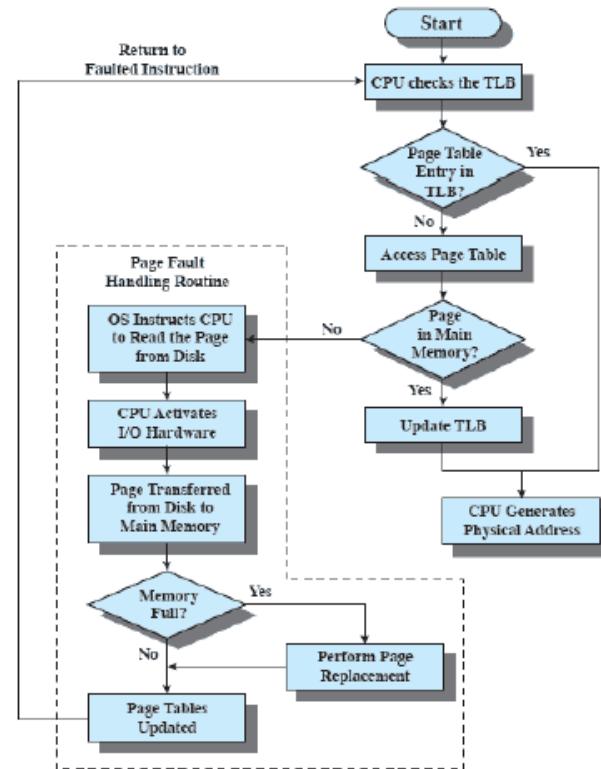
TLB – Translation Lookaside Buffer je HW komponenta u okviru MMU za transformisanje – prevođenje VIRTUALNIH u FIZIČKE adrese bez pristupa tabeli stranica. Ja mislim da ne dolazi do smanjenja grešaka stranica. Zašto: Greška stranice znači da smo pristupili Tabeli stranica i ustanovili da za stranicu koja je nama potrebna bit prisutnosti nije postavljen, tj. Ta stranica nije u glavnoj memoriji i moramo da je pribavimo. E sad, ako ja iskoristim TLB ja ću samo imati fizičku adresu stranice koja je nedavno prevedena, ali ako hoću stranice koje nisu u glavnoj memoriji džabe mi TLB, dolazi do promašaja. TLB samo ubrzava pretraživanje, jer se koristi asocijativno preslikavanje, odnosno moguće je pretražiti ceo TLB jednovremeno zahvaljujući posebnom HW, ali ne smanjujem broj grešaka tako. Barem ja tako mislim.

Tačno, ubrzava prevođenje, ali ne smanjuje broj grešaka.



Algoritam prevodenja adresa

- Algoritam prevodenja adresa korišćenjem TLB i tabele stranica



Virtuelna memorija
Operativni sistemi

Prof. dr Dragan Stojanović

14. 2016 - JAN

Navesti osnovne razlike algoritama zamene stranica Clock sa jednim i dve kazaljke. Pretpostaviti da sledeća tabela stranica prikazuje trenutnu zauzetost okvira stranicama, tj. okvir 0 trenutno sadrži stranicu 17, okvir 1 stranicu 32, itd. Pretpostaviti da se koristi Algoritam sata (Clock) kao strategija zamene stranica i bit upotrebe se postavlja na 1 kada se stranica umesti u memoriju. Treci vrsta tabele prikazuje trenutnu vrednost bitova upotrebe.

Stranica okvir	0	1	2	3	4	5	6	7
Stranica	17	32	41	5	7	13	2	20
bit upotrebe	1	0	0	0	0	1	1	0

Ukoliko pokazivač sata pokazuje na okvir 4 koji stranice će biti zamjenjene sledećom nizom stranica zauzeta: 32, 14, 15, 2, 18, 14. Objasni razlozi! Prikazati sadržaj memorije (okvira) i bitova korišćenja.

15. 2015 - APR

Razmotrite operativni sistem koji koristi strančenje u dva nivoa (direktorijum i tabelu stranica) i TLB. Vreme pristupa TLB-u je 10ns (za pogodak i promatranje), a procenat pogodaka je 99%. Vreme pristupa memoriji je 100ns. Prosečno 0.001% svih memorijskih referenci generiše grešku stranica za čije rukovanje OS-u je neophodno 3ms. Koliko je efektivno vreme pristupa memoriji? Objasni!

$$p=0.001=10^{-3}$$

$$EFP = (1-p) \cdot \text{Pogodak} + p \cdot \text{Promasaj}$$

$$\text{pogodak} = (10\text{ns} + 100\text{ns})$$

Promasaj=(10ns+100ns+100ns+3ms)

Zasto u promasaju 2×100 ns ? Ako je tlb miss znaci imamo 2 referenciranja memorije ako je u tlb sve sto ti treba imas 1 referenciranje manje ja bar mislim

16. 2015 - MAR

Koja je razlika između planiranja bez prekidanja (*non-preemptive*) i planiranja sa prekidanjem (*preemptive*) i kada se obavlja prekidanje? Razmatraju sistem sa 3 stacionarnim okvira i sledećim sekvencom:

5 6 9 | 3 5 6 | 3 6 9 | 4 3 9 | 6 4 9

Koliko grešaka stranica bi bilo generisano ako bi se koristio a) FIFO, b) LRU i c) optimalni algoritam zamene stranica. Da li bi se smatratio broj grešaka stranica ako bi se koristio TLB i na koji način?

17. 2015 - JAN

Navesti osnovne razlike algoritama zamene stranica Clock sa jednom i dve „kazaljke“. Prepostaviti da sledeća tabela stranica prikazuje trenutnu zauzetost okvira stranicama, tj. okvir 0 trenutno sadrži stranicu 12, okvir 1 stranicu 10, itd. Prepostaviti da se koristi Algoritam sata (Clock) kao strategija zamene stranica i bit upotrebe se postavlja na 1 kada se stranica smesti u memoriju. Treća vrsta tabele prikazuje trenutnu vrednost bitova upotrebe.

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

Ukoliko pokazivač sata pokazuje na okvir 4 koje stranice će biti zamenjene sledećim nizom straničnih zahteva: 10, 14, 15, 2, 18, 14, 2. Objasniti zašto! Prikazati sadžaj memorije (okvira) i bitova korišćenja.

III-treca

1. 2018 - OKT

- a) U kojim slučajevima OS vrši prekidanje (*preemption*) procesa.
b) Kojim procesima/nitima savremeni OS (Windows, MacOS, Linux,...) daju prednost pri raspoređivanju, orijentisanim na CPU ili na U/I i na koji način?

- a) 3. pitanje pod a i 2. pod a

Odgovor na celo je u 4. pitanju

2. 2018 - SEP

(a) Šta znači „prekidanje“ u planiranju sa prekidanjem (*preemptive scheduling*) i kada se prekidanje dešava?

(b) Koji od navedenih algoritama su sa prekidanjem: FCFS, RR, SPN (*Shortest Remaining Time*), Multilevel Feedback ?, SRT

a)

OS može prekinuti proces koji se izvršava i premestiti ga u red spremnih procesa
Omogućava bolje usluge jer nijedan proces ne može dugo monopolizovati procesor

Prekidanje predstavlja odluku OS da prekine proces koji se trenutno izvršava u situacijama kada dođe novi proces, desni se prekid koji stavlja blokirani proces u red spremnih ili periodično na osnovu prekida iz generatora takta.

b)RR,SRT,Feedback

Razlika između planiranja (preemptive) sa i bez prekidanja (non-preemptive). Ostatak pitanja je standardno januar 2017.

Režim izbora određuje trenutke u vremenu u kojima se izvršava funkcija izbora. Postoje 2 opšte kategorije:

Bez prekidanja – Proces je u stanju izvršavanja sve dok se ne završi, ili se sam ne blokira da bi čekao na UI ili da bi zahtevao neku uslugu OS

Sa prekidanjem – Proces koji se izvršava može da se prekine i smesti u stanje spremjanja.

Odluka o prekidanju se donosi kada dođe novi proces, ili kada se desi prekid koji stavlja blokirani proces u red spremnih ili periodično na osnovu prekida iz generatora takta.

3. 2018 - JAN

- a) Šta je prekidanje (*preemption*) i u kojim slučajevim se javlja prilikom izvršenja procesa.
b) Kojim procesima/nitima savremeni OS (Windows, MacOS, Linux,...) daju prednost u izvršavanju, orijentisanim na CPU ili na U/I i na koji način?

a) kada dođe novi proces, ili kada se desi prekid koji stavlja blokirani proces u red spremnih ili periodično na osnovu prekida iz generatora takta

4. 2017- OKT

Odgovoriti i objasniti ukratko.

- Kojim procesima/nitima savremeni OS (Windows, MacOS, Linux,...) daju prednost u izvršavanju, orijentisanim na CPU ili na U/I i na koji način?
- Koja je prednost, a koji nedostatak malog kvanta za izvršenje?
- Koja je svrha različite dužine kvanta za različite redove u Multi-level feedback algoritmu planiranja procesa?

Prednost uvek imaju U/I orijentisani procesi/niti, zato što će oni kratko zadržati procesor i moći će da čekaju na akciju korisnika. Prioritet je da korisnik ne ispašta. A da prioritet imaju CPU orijentisani procesi onda bi korisnik morao uvek mnogo da čeka.

Prednosti i nedostaci malog kvanta izvršenja?

Ne mogu da nađem tačno gde sam pročitao, ali naći ću pa tačno da prepišem, ali pamtim poentu: Kad je kraći vremenski kvant onda je bolje iskorišćen procesor i svi procesi će biti opsluženi barem na kratko. Ipak, nedostak je često menjanje procesa i to zahteva učešće OS.

Različite dužine vremenskog kvanta u MultiLevel Feedback-u?

SVAKI od nivoa ima određeni prioritet, nivo 1 je najmanjeg prioriteta. Što je dalji nivo, dalji od jedinice, to će mu se kasnije pristipiti jer se nivou n pristupa tek kad je nivo n – 1 prazan, ali zato mu se dodeljuje veći vremenski kvant da se na taj način kompenzuje kasnije pristupanje, a i pošto se nalazi u daljem nivou znači da mu je istekao vremenski kvant koji mu je prethodno dodeljen i prepostavlja se da će mu biti potrebno više vremena da se izvrši.

//Kraci odgovor

- b) Prednost je da U/I orijentisani procesi neće gladovati, nedostatak CPU orijentisani ili dugi procesi će gladotvati, vreme zadržavanja dugih procesa je veoma dugo i dolazi do cestog smenivanja procesa na CPU
- c) Sa fiksnom dužinom kvanta, vreme zadržavanja dugih procesa može biti alarmantno. Da bi se kompenzovao ovaj nedostatak sa nivoom reda uvećava se vremenski kvant

5. 2017-APR

Koji je cilj algoritama SJF i SRTF u planiranju procesa (niti) i zašto ih je teško implementirati na realnom OS? Navesti i opisati alternativni pristup (algoritam) sa istim ciljem koji se može lako implementirati u realnom OS. Za sledeće procese odredi redosled izvršenja procesa, i za svaki algoritam prosečno vreme zadržavanja (turnaround time) i vreme čekanja. A) FIFO, b) SJF, c) RR (vremenski kvant =1). Novi process se dodaje na kraj reda spremnih.

Process ID	Arrival Time	Expected CPU Running Time
Process 1	0	4
Process 2	2	5
Process 3	3	3
Process 4	8	4

Prepostavljam da je SJF Shortest Job First, tj. Shortest Process Next. Neka je tako. E ako je tako onda su SJF I SRTF(shortest remainig time) isti. Bira se proces sa najkraćim očekivanim vremenom obrade. Jedina razlika je u tome što kod SJF nisu dozvoljeni prekidi, a kod SRTF jesu. Teško ih je implementirati zato što nije lako odrediti koliko vremena će biti potrebno procesu za izvršenje. "Ne može se predvideti budućnost". FCFS i RR su laci za implementaciju, a svi od algoritama u ovoj grupi imaju isti cilj, tako da može bilo koji, ali ovi su mi najlakši za implementaciju, ne znam da li se misli na neki konkretni.

- a) FIFO
- | | | | |
|-----|-----|------|-------|
| 0-4 | 4-9 | 9-12 | 12-16 |
| P1 | P2 | P3 | P4 |

6. 2017- JAN

3. a) Kom procesu će Windows/UNIX/Linux OS dati prednost pri planiranju procesa algoritma planiranja:
P1: prethodno deblokiran jer je sadržaj datoteke koji je zahtevao učitan u RAM, P2: prethodno istrošio vremenski kvant određujući sledeći potez šahovskog algoritma? Ukratko objasniti kako.
b) Za sledeći skup procesa prikazati redosled izvršenja procesa i prosečno vreme zadržavanja/prolaska (*turnaround time*) ukoliko se koriste algoritmi 1. RR (q=2) 2. SRT (Shortest Remaining Time) 3. ML Feedback q=2¹.

Proces	Vreme izvršenja	Vreme aktiviranja
A	6	0
B	10	3
C	4	6
D	7	9
E	5	12

- a) Trebalo bi p1 jer os daje prednost ui orijentisanim procesima. Proveri sa profesorom!
b) <https://www.youtube.com/watch?v=1fFgy1kjag/> /feedBack
<https://www.youtube.com/watch?v=aWIQYIIBZDs> //roundRobin

7. 2016 - OKT2

8. Koji su osnovni principi planiranja CPU u više redova (*Multi-Level Queues*) koji su implementirani u Unix i Windows operativnim sistemima? Objasniti. Razmotriti sledeće procese:

Proces	Vreme izvršenja	Prioritet	Vreme aktiviranja
P1	50 ms	4	0 ms
P2	20 ms	1	20 ms
P3	100 ms	3	40 ms
P4	40 ms	2	60 ms

Prikazati redosled izvršenja procesa ukoliko se koriste sledeći algoritmi: (a) SRT (Shortest Remaining Time), (b) Prioritet sa prekidanjem (preemptive) (manji broj znači veći prioritet), (c) Round Robin sa vremenskim kvantom 30 ms.

u suštini to je uvek planiranje po prioritetu sa više redova

U redovima se nalaze samo spremni procesi koji su u memoriji

Prioritet procesa se preračunava svake sekunde

Planer pretražuje redove počev od reda najvišeg prioriteta i bira prvi proces iz prvog nepraznog reda. Dodeljuje mu CPU na 1 kvant, nakon čega ga prekida, osim ako se proces nije u međuvremenu blokirao, i smešta na kraj reda odakle je izabran (RR planiranje)

Za razliku od feedback-a ovde su svi redovi round robin!

**(nije za ovo pitanje: kod feedBacka je samo poslednji red roundRobin, a ostali su fifo)

8. 2016 - OKT

3. Razmotriti sledeće procese:

Proces	Vreme izvršenja	Prioritet	Vreme aktiviranja
P1	70 ms	3	0 ms
P2	120 ms	1	10 ms
P3	20 ms	4	20 ms
P4	50 ms	2	30 ms

Objasniti osnovnu logiku algoritma za planiranje procesa i prikazati redosled izvršenja procesa ukoliko se koriste algoritmi a) SRT b) Prioritet sa prekidanjem (*preemptive*) (manji broj znači veći prioritet) i c) Round Robin sa vremenskim kvantom 20 ms.

9. 2016 - SEP

3. Sledeci procesi treba da budu izvršeni na jednoprocесорском систему

PID	Vreme startovanja	Vreme izvršenja (CPU burst)
1	0	4
2	1	3
3	3	2
4	10	6
5	12	5

Pretpostaviti da je vremenski period (overhead) za obavljanje *context switch*-a jedna vremenska jedinica. Za svaki od navedenih metoda za planiranje CPU prikazati: (a) vremenski dijagram izvršavanja procesa (ii) prosečno vreme zadržavanja (turnaround).

- a) FCFS – (*First Come First Served*)
- b) SPN (*Shortest Process Next*)
- c) SRTN – (*Shortest Remaining Time Next*)
- d) RR (*Round Robin*), vremenski kvant = 3
- e) Multilevel Feedback Queue sa redovima 1-4, vremenski kvant = 2², gde je i broj reda i procesi su inicijalno smješteni u prvi red (nivo 1)

10. 2016 - JUN

OS koristi planiranje procesa u više redova (*Multi-Level Feedback*) sa neograničenim brojem redova. Novi process startuje na nivou 0. Za red i vremenski kvant iznosи 2¹. Inicijalno postoji samo jedan proces u sistemu koji se izvršava 9 vremenskih jedinica pre završetka. Na svake dve vremenske jedinice on kreira novi proces. Svaki novi process se izvršava jedan vremenski kvant pre završetka. Kreiranje procesa ne generiše prekidanje tekućeg procesa. Nijedan od procesa ne izvršava neki blokirajući sistemski poziv, izvršavaju se sve do prekidanja ili završetka. Koje je vreme zadržavanja inicijalnog procesa? Načrtati i objasniti vremenski dijagram izvršenja procesa.

11. 2016 - APR

Koja je razlika između planiranja bez prekidanja (*non-preemptive*) i planiranja sa prekidanjem (*preemptive*)? Pet procesa, identifikatora od A do E započinju izvršavanje u istom trenutku na računarskom sistemu. Njihova vremena izvršenja su repsektivno 10, 6, 2, 4 i 8 vremenskih jedinica. Njihovi prioriteti su 3, 5, 2, 1, i 4, pri čemu je 5 najveći prioritet. Za svaki od algoritama planiranja odrediti i objasniti vremenski dijagram izvršenja procesa, kao i srednje vreme zadržavanja/prolaska (turnaround time): a) round robin, b) prioritet bez prekidanja c) prvo najkraći posao (*shortest job first*)

- a) fali vremenski kvant i kako se menjaju na cpu posto svi pocinju u istom trenutku?, proveri i za c)

12. 2016 - MAR

3. Opisati osnovne principe algoritama za planiranje procesa: SRT (*Shortest Remaining Time*), Prioritet sa i bez prekidanja i Round Robin. Razmotriti sledeće procese:

Proces	Vreme izvršenja	Prioritet	Vreme aktiviranja
P1	70 ms	3	0 ms
P2	120 ms	1	10 ms
P3	20 ms	4	20 ms
P4	50 ms	2	30 ms

Prikazati redosled izvršenja procesa ukoliko se koriste algoritmi a) SRT b) Prioritet sa prekidanjem (*preemptive*) (manji broj znači veći prioritet) i c) Round Robin sa vremenskim kvantom **20 ms**, pri čemu vreme za obavljanje *context switch-a* iznosi **5 ms**.

13. 2016 - JAN

3. a) Kojim procesima OS daje prednost pri planiranju procesa korišćenjem (*Multi-Level Feedback*) algoritma: orijentisanim ka CPU ili I/O? Ukratko objasni kako!
b) Za sledeći skup procesa prikazati redosled izvršenja procesa ukoliko se koriste algoritmi i) RR (q=1) ii) RR (q=4) iii) ML Feedback q=1 iv) ML Feedback q=2.

Proces	Vreme izvršenja	Vreme aktiviranja
A	3	0
B	5	1
C	2	3
D	5	9
E	5	12

14. 2015 - APR

3. Razmotri sledeće procese:

Proces	Vreme izvršenja	Prioritet	Vreme aktiviranja
P1	50 ms	4	0 ms
P2	20 ms	1	20 ms
P3	100 ms	3	40 ms
P4	40 ms	2	60 ms

Prikazati redosled izvršenja procesa ukoliko se koriste algoritmi a) SRT b) Prioritet bez prekidanja (*non-preemptive*) (manji broj znači veći prioritet) i c) Round Robin sa vremenskim kvantom 30 ms. Koliko je prosečno vreme čekanja za sve ove algoritme planiranja?

15. 2015 - MAR

3. Opisati osnovne principe algoritama za planiranje procesa: SRT (Shortest Remaining Time), Prioritet sa i bez prekidanja i Round Robin. Razmotriti sledeće procese:

Proces	Vreme izvršenja	Prioritet	Vreme aktiviranja
P1	70 ms	3	0 ms
P2	120 ms	1	10 ms
P3	20 ms	4	20 ms
P4	50 ms	2	30 ms

Prikazati redosled izvršenja procesa ukoliko se koriste a) SRT b) Prioritet sa prekidanjem (preemptivne) (manji broj znači veći prioritet) i c) Round Robin sa vremenskim kvantom 20 ms.

16. 2015 - JAN

- a) Kojim procesima OS daje prednost pri planiranju: orientisanim ka CPU ili U/I? Ukratko objasniti generani princip implementiran u savremenim OS?
- b) OS koristi planiranje procesa u više redova (*Multi-Level Feedback*) sa 5 redova. Novi proces startuje na nivou 0. Za red i vremenski kvant iznosi 2^i . Inicijalno postoji samo jedan proces u sistemu koji se izvršava 14 vremenskih jedinica pre završetka. Na svake tri vremenske jedinice on kreira novi proces. Svaki novi proces se izvršava dva vremenska kvanta pre završetka. Kreiranje procesa ne generiše prekidanje tekućeg procesa. Nijedan od procesa ne izvršava neki blokirajući sistemski poziv, izvršavaju se sve do prekidanja ili završetka. Koje je vreme zadržavanja inicijalnog procesa? Načrtati i objasniti vremenski dijagram izvršenja procesa.

IV-cetvrt

1. 2018 -OKT

Zašto je potrebno da se u okviru operativnog sistema implementira U/I softver nezavistan od U/I uređaja?
Da li drajver uređaja spada u ovaj softver i zašto?

<https://flylib.com/books/en/3.275.1.28/1/> stavka 3.2.4 sa linka

It should be possible to write programs that can access any I/O device without having to specify the device in advance. For example, a program that reads a file as input should be able to read a file on a floppy disk, on a hard disk, or on a CD-ROM, without having to modify the program for each different device.

Trebalo bi biti moguće pisati programe koji mogu pristupiti bilo kojem I / O uređaju, a da prethodno ne moraju da ga specificiraju. Na primer, program koji čita datoteku kao ulaz trebalo bi da može da čita datoteku na disketu, na tvrdom disku ili na CD-ROM-u, bez potrebe da menjate program za svaki drugi uređaj.

Osn fja ovakvih softvera je da izvrsi u fje koje su zajednicke svim uređajima i da obezbedi jedinstven interfejs ka korisnickim softverima.

Fje ui softvera nezavisnih od uređaja:

-Jedinstven interfejs za drajvere uređaja

-Baferovanje

-Prijava gresaka

-Dodeljivanje/Oslobadjanje odgovarajucih uređaja

-Obezbedjivanje velicine bloka nezavisnog od uređaja

Drajver uređaja spada u UI softver zavistan od uređaja.

Zato sto se on implementira kako bi upravljao funkcionisanjem specificnog uredjaja ili klase istih uredjaja.

//Pomocu drajvera se vrši komunikacija sa kontrolerom uredjaja

3. 2018 - JAN

- a) Linux file sistem ima blokove veličine 1KB i disk adrese od 4B. Kolika je maksimalna veličina datoteke ukoliko se u i-čvoru smešta 12 direktnih blokova, kao i adrese jednostrukog, dvostrukog i trostrukog indirektnog bloka?
- b) Napisati pseudo-kod za upis u N-ti bloka datoteke u UNIX file sistem-u.

Linux

Mislim da je isto kao 2017 jan b i c

4. 2017- OKT

Fajl sistem upravlja smeštanjem podataka na disk kapaciteta 32 bloka od 1024 bajta. Trenutno su svi blokovi diska slobodni (disk je prazan). Za upravljanje slobodnim prostorom fajl sistem koristi bit mapu. Fajl sistemu je stigao zahtev od procesa P8 za upis fajla STUDENTI dužine 9 blokova, a od procesa P2 za upis fajla OCENE dužine 3 bloka. Oba fajla se nalaze u direktorijumu FAKULTET. Gde će fajl sistem smestiti ove fajlove ako za dodelu prostora na disku koristi: (a) kontinualnu dodelu, (b) FAT tabelu i (c) indeksiranje? Prikažite sadržaj direktorijuma i dodeljene blokove fajlovima STUDENTI i OCENE za sva tri algoritma dodele prostora na disku. Takođe prikažite sadržaj bit mape pre i posle dodele prostora za fajlove STUDENTI i OCENE. Gde fajl sistem čuva bit mapu, a gde direktorijum?

- a) Kontinualnu dodelu
- b) FAT tabelu
- c) Indeksiranje

Prikažite sadržaj direktorijuma i dodeljene blokove fajlovima STUDENTI i OCENE za sva 3 algoritma dodele prostora na disku. Takođe prikažite sadržaj bit mape pre i posle dodele prostora za fajlove STUDENTI i OCENE. Gde fajl sistem čuva bit mapu, a gde direktorijum?

Bit mapa = Bit vektor

Bit vektor se čuva u slobodnim blokovima na disku. Bit mapa bi trebalo biti na disku, možda bi trebalo izabrati blok 0 da se tu smesti bit mapa.

Gde se čuva direktorijum? Disk.

Traženje slobodnog prostora od n blokova se svodi na nalaženje sekvene od n nula u bit vektoru.

- a) Kontinualna dodata – fajl STUDENTI će biti smešten od bloka sa indeksom 0 zaključno sa indeksom 8, a fajl OCENE od 9 zaključno sa 11.

Tada će u bit mapi na prvih 12 mesta biti 1-ce, dok će ostalo biti 0.

Direktorijum Fakultet:

Naziv datoteke	Blok početka	Dužina
Studenti	0	9
Ocene	9	3

- b) FAT tabela – "Izbor blokova je sada jednostavna stvar – svaki slobodan blok može da se doda lancu." Možemo da izaberemo blokove proizvoljno jer je disk prazan. Najbolje bi bilo da idu sekvensijalno i pošto imamo bit mapu mislim da je to i nalogičnije. Direktorijum ponovo sadrži startni blok i dužnu. Tačno. Samo je sadržaj FAT-a takav da je FAT od 0 na 1, od 1 na 2 i tako dalje.
- c) Indeksiranje – Analogno postupku pod b) s razlikom što će direktorijum sada izledati ovako pod pretpostavkom da smo zauzeli prvih 9 pa 3 bloka. Da se u bloku 9 nalazi indeksni blok za Studente i u bloku 13 ind.blok za Ocene.
- Bit mapa na prvih 14 mesta ima 1-ce pa onda 0.

Direktorijum:

Datoteka	Indeksni blok
Studenti	9
Ocene	13

5. 2017- APR

Objasniti zašto je korišćenje FAT tabele u Windows XP za evidentiranje slobodnih blokova pouzdanije i boljih performansi u odnosu na listu slobodnih blokova. UNIX file sistem je upravo rebusovan, te su baferi i keš obrisani. Koji je minimalan broj disk blokova koji mora biti pročitan da bi se pročitao 2^{15} bajt datoteke /os/ispit/april17, ukoliko file sistem koristi blokove od 1KB, 10 direktnih blokova i i-nodu i 32-bitne adrese blokova i brojeve i-noda? Direktorijum os je veličine 20KB, dok je direktorijum ispit veličine 7KB. Objasniti.

(Upravljanje datotekama / slajd 91)

2^{15} B kad podelimo PO 2^{10} je 32. blok datoteke, znači nije među direktnim nego među jednostrukim indirektnim.

U indirektnom imamo 32b adrese, znači 256 blokova u indirektnom, nama treba 32. blok.

Knjiga, strana 564. kaže sledeće: U sistemu UnixV , dužina bloka je 1KB, a svaki može da sadrži ukupno 256 adresa blokova.

Direktorijum os je 20KB pa ima 20 blokova, a ispit 7 blokova. Ako je root direktorijum u glavnoj memoriji onda imamo da pročitamo jedan i-čvor os-a, to je jedan blok, a onda treba da pročitamo njegovih 20 blokova. Ali pošto je 20 blokova, onda je prvih 10 direktnih blokova, jedan indirektni i još 10. Kad vidimo tu sadržaj nađemo broj foldera datoteke ispit, to je plus 1. Svih sedam blokova je direktnih, tu vidimo datoteku april, pročitamo i-čvor i da bi pročitali i-čvor moramo da pročitamo jednostruki indirektni i 22. blok u njemu.

To treba sabrati: $(1 + 10 + 1 + 10) + (1 + 7) + (1 + 1 + 22)$

6. 2017- JAN

- a) Koja je prednost evidencije blokova datoteke ulančavanjem pomoću FAT u odnosu na klasično ulančavanje blokova datoteke? Objasniti ukratko.
- b) UNIX file sistem ima blokove veličine 4KB i disk adrese od 4B. Kolika je maksimalna veličina datoteke ukoliko se u i-čvoru smešta 10 direktnih blokova, kao i adrese jednostrukog, dvostrukog i trostrukog indirektnog bloka?
- c) Napisati pseudo-kod za čitanje N-tog bloka datoteke.

a) Prednosti su što se sada ceo blok koristi samo za podatke, ne i za pokazivače i brži je random pristup jer tačno znamo koji blok ukazuje na taj konkretni random da tako kažem. Jer kod klasičnog ulančavanja treba da se učita blok da bi se uzela adresa n-tog bloka, a ovde imamo FAT u memoriji pa u memoriji nađemo adresu n-tog bloka što je, naravno, mnogo brže.

b) Veličina bloka = 4KB

Disk adresa = 4B

PROVERI DEO: Veličina bloka / disk adresa = broj adresa, odnosno $2^{12} / 2^2 = 2^{10} = 1024$ adresa ima u jednom bloku. Tačno je.

Nivo	Broj blokova	Broj bajtova
Direktan	10	$10 \times 4KB = 40KB$
Jednostruki indirektan	1024	$1024 \times 4KB = 4MB$
Dvostruki indirektan	$1024 \times 1024 = 1M$	$1M \times 4KB = 4 GB$
Trostruki indirektan	$1024 \times 1M = 1G$	$1G \times 4KB = 4TB$

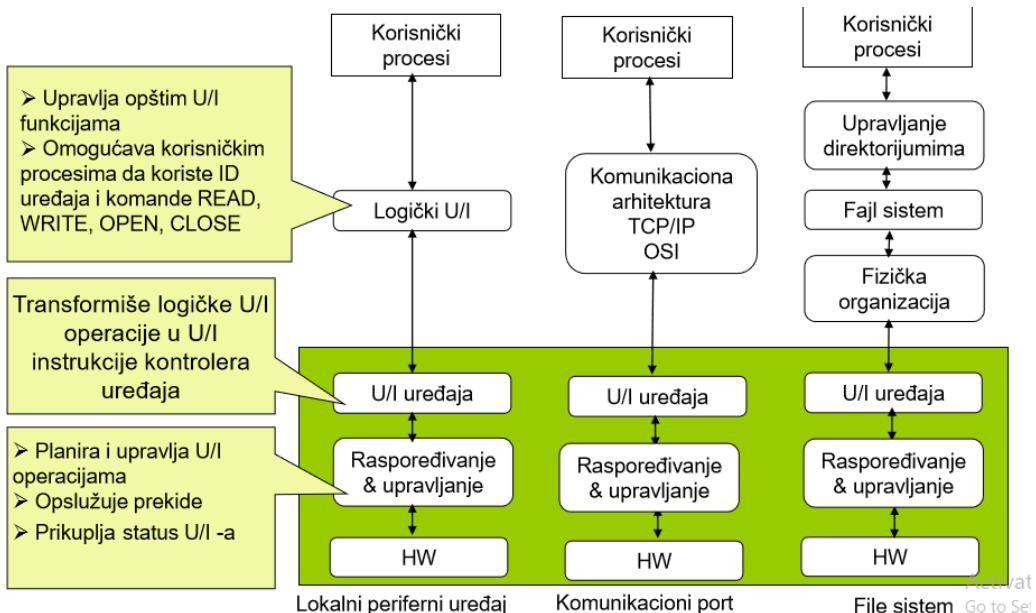
Maksimalna veličina datoteke u ovoj šemi je preko 4TB.

c)

```
If (N < 10)
    Return read(blok[N]);
Else if (N < 1024 + 10)
    Jednostruki = read(blok[11]);
    Return read(jednostruki[ N - 10 ])
Else if (N < 1024 x 1024 + 1024 + 10)
    Dvostruki = read(blok[12])
    Return read(dvostruki[N - 1024 - 10]);
```

7. 2016 - OKT2

Opisati hijerarhijsku organizaciju UI sistema. Objasniti proces baferovanja i navesti u kom sloju se obavlja baferovanje. Objasniti zašto se očekuje poboljšanje performansi upotrebom dvostrukog u odnosu na jednostruko baferovanje.



Lokalni periferijski uređaji

- Logički U/I: – Upravlja uređajem kao logičkim resursom
- U/I uređaja: – Transformiše logičke U/I operacije u sekvencu U/I instrukcija kontrolera uređaja
- Planiranje i upravljanje: – Izvršava uređenje i planiranje U/I operacija i upravljanje operacijama. Rukovanje izuzecima i očitavanje i prijavljivanje U/I statusa se obavljaju na ovom sloju.

Komunikacioni port

- Idenično kao u prethodnom slučaju samo što je logički U/I modul zamenjen komunikacionom arhitekturom – Ova arhitektura se sastoji od više slojeva – Primer, TCP/IP

File sistem

- Upravljanje direktorijumima – Na ovom nivou simbolička imena datoteka se prevode u identifikatore kojima se referenciraju datoteke, i obezbeđuje podrška za korisničke operacije nad direktorijumima, poput add , delete , reorganize
- File sistem – Obezbeđuje logičku strukturu datoteka, korisničke operacije, poput open , close , read , write , kao i prava pristupa
- Fizička organizacija – Logičke reference na datoteke se prevode u fizičke adrese na sekundarnoj memoriji, vodeći računa o fizičkoj strukturi medijuma

-Procesi moraju da čekaju dok se ne završi U/I i tek kada su podaci raspoloživi mogu da nastave sa izvršenjem

-Da bi se izbeglo uzajamno blokiranje neke stranice procesa moraju biti ("zaključane") u glavnoj memoriji tokom izvršavanja U/I

Bafer je oblast u memoriji gde se smeštaju podaci dok se prenose između U/I uređaja i memoriskog prostora procesa – Kada se ova U/I operacija izvodi može se garantovati da je zapisano stanje podataka u trenutku poziva U/I operacije

Efikasnije je da se transfer ulaznih podataka obavi unapred u odnosu na zahtev, a da se transfer izlaznih podataka izvrši sa zakašnjenjem u odnosu na zahtev

//U kom sloju? Transportnom slovju----Nijedan od ovih se ne zove transportni?!

Dva bafera omogucavaju da proces može prenositi podatke u ili iz jednog bafera dok operativni sistem prazni ili puni drugi bafer;nema cekanja na ui kao kod jednostrukog (If the I/O operation involves the same disk that is used for swapping, it hardly makes sense to queue disk writes to the same device for swapping the process out. This attempt to swap the process and release main memory will itself not begin until after the I/O operation finishes, at which time swapping the process to disk may no longer be appropriate)

8. 2016 - OKT

Objasniti zašto je korišćenje bit vektora za evidentiranje slobodnih blokova pouzdanije i boljih performansi u odnosu na listu slobodnih blokova. UNIX file sistem je upravo rebutovan, te su baferi i keš obrisani. Koji je minimalan broj disk blokova koji mora biti pročitan da bi se pročitao milioniti bajt datoteke /a/big/file, ukoliko file sistem koristi blokove od 4KB i 32-bitne adrese blokova i brojeve i-noda? Objasniti.

Cim su fajlovi verovatno je zadnja

9. 2016 - SEP

Opisati osnovne karakteristike algoritama za planiranje disku FCFS, SSF, SCAN i C-SCAN. Disk ima 200 trasa, numerisanih 0 do 199 i u redu zahteva se nalaze zahtevi za trasama 86, 147, 91, 177, 94, 153, 102, 175, 132. Prepostavimo da je glava diska upravo završila pristup trasi 125 i da trenutno pristupa trasi 143. Prikažite sekvencu u kojoj će se zahtevi obradivati, kao i ukupan broj predenih staza za svaku od sledećih politika raspoređivanja: FCFS, SSF, SCAN i C-SCAN

FCFS(FirstComeFirstServed)

- Prvi došao prvi uslužen
- Zahtevi se obrađuju sekvencijalno
- Fer za sve procese

SSF(ShortestSeekFirst)

- Uvek se bira minimalno vreme traženja
- Bira se ui zahtev koji zauzima najmanje kretanje glave diska sa njegove trenutne pozicije

SCAN(Elevator)

- Glava se pomera u jednom pravcu zadovoljavajuci zahteve sve dok ne dostigne poslednju trasu u tom pravcu.Tada se se glava okreće i pomera se u obrnutom pravcu ,nastavljajuci da usluzuje zahteve na koje naidje

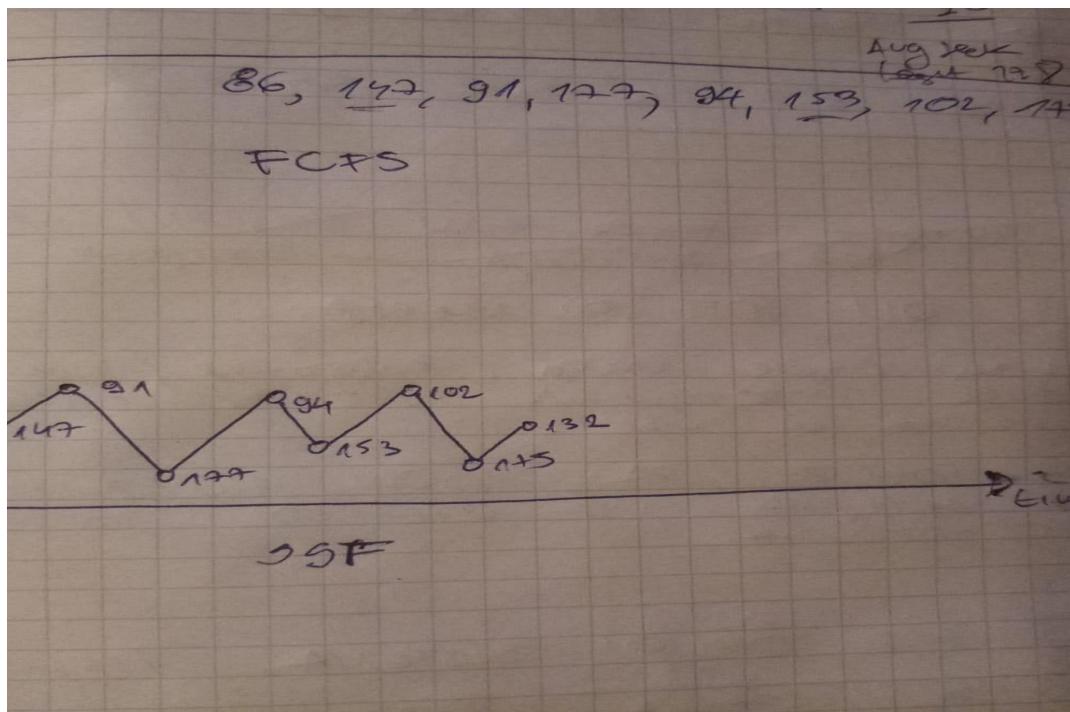
CSCAN

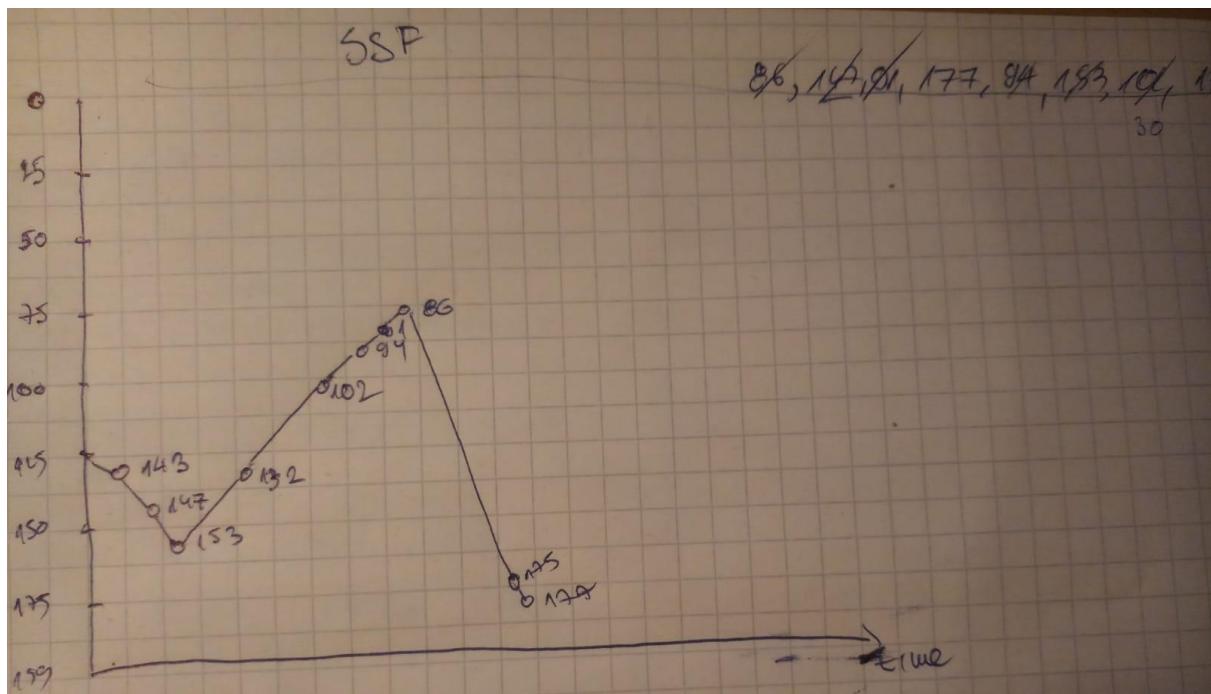
- Ogranicava se skaniranje(postoji i ta rec) na jedan pravac

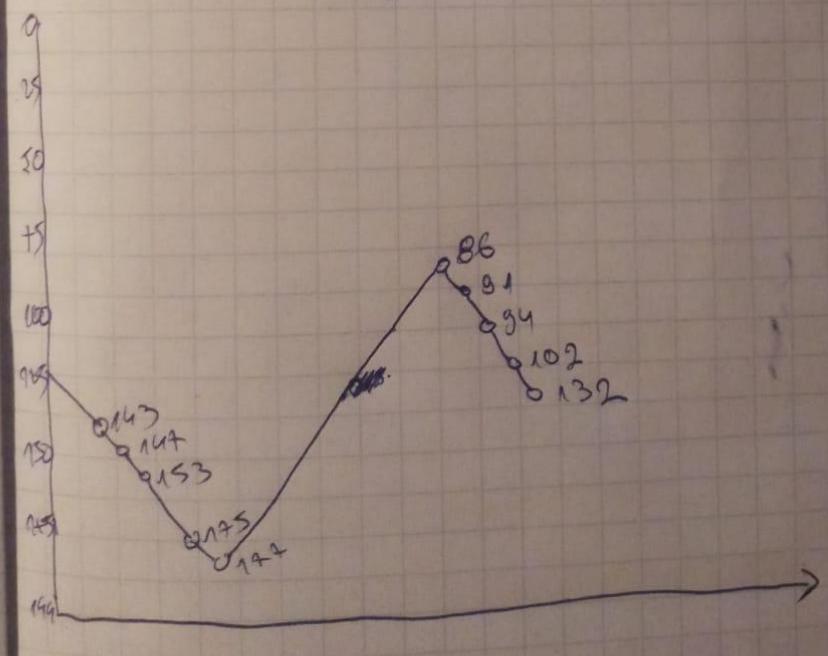
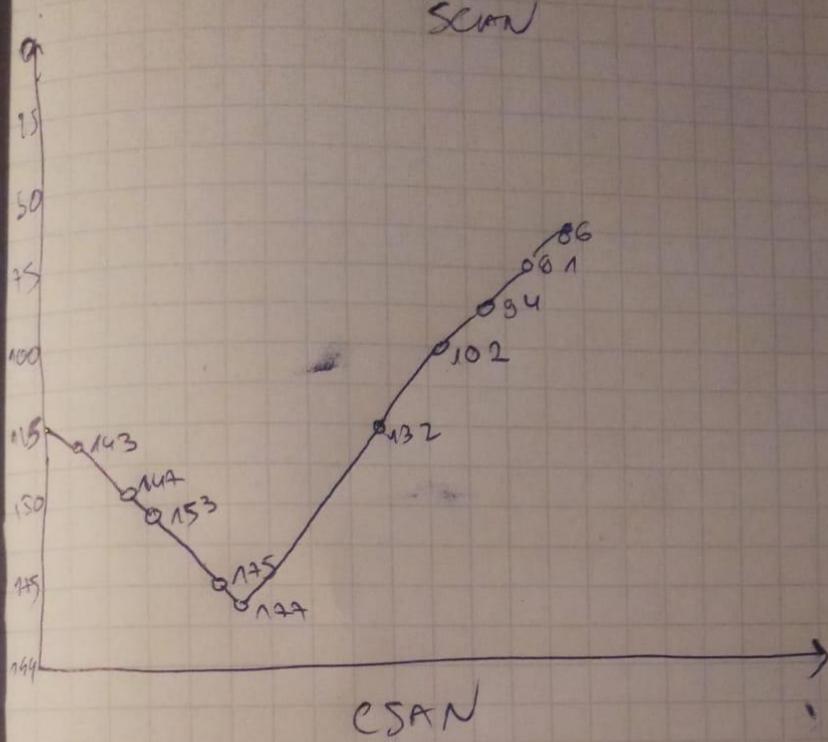
- Kada se dostigne zadnja trasa u jednom pravcu ili kada više nema zahteva u tom pravcu, glava se vraća na drugi kraj diska i tada skaniranje ponovo startuje

Fifo-125, 143, 86, 147, 91, 177, 94, 153, 102, 175, 132

18+57+61+56+86+83+59+51+73+43/11







10. 2016 - JUN

Pretpostaviti da datoteka *bar* sadrži blokove u sledećem redosledu: 2,5,3,4,8,1,6, 7

- a. Prikazati strukturu direktorijuma i strukturu diska sa blokovima (0-15), kao i internu strukturu bloka ako se koristi dodela indeksiranjem u grupama blokova, pri čemu je blok 9 indeksni.
- b. Prikazati strukturu direktorijuma i strukturu diska sa blokovima (0-15), ako se koristi dodela ulančavanjem pomoću tabele u memoriji.
- c. Navesti i objasniti koliko disk operacija je neophodno da se u glavnu memoriju učita sadržaj datoteke *usr/courses/os/ispit.pdf* ukoliko je samo i-čvor root direktorijuma u glavnoj memoriji, a veličina svih direktorijuma na putu (*path*) i same datoteke je manja od veličine bloka. Nakon završetka read naredbe, koliko je disk operacija neophodno za čitanje datoteke *seminarski.txt* sa istog direktorijuma. Objasniti

Zadnja

11. 2016 - APR

Navesti razliku između drajvera uređaja i kontrolera uređaja i opisati njihove osnovne funkcije. Razmotriti sistem sa diskom sa 8 sektora po stazi i 512 B po sektoru. Disk se rotira brzinom od 3000rpm i ima prosečno vreme traženja (seek time) od 15ms. Posmatrati datoteku koja ima 8 blokova pri čemu 1 blok zauzima 1 sektor i izračunati ukupno vreme neophodno za pristup celoj datoteci ako se koriste metode alokacije a)kontinualna alokacija i b) indeksiranje

8 sektora

1 sektor = 512B

r = 3000 rpm / 60= 500rps

T_s = 15ms

1 sektor = 1 blok

T_a – vreme pristupa datoteci

a) T_a = T_s + 1/2r + b/rN = 15ms + 1ms + vreme čitanja 8 sektora

Vreme čitanja osam sektora je jednako čitanju 1 cele staze, odnosno: N / r x N = 1 / r = 2ms ---->1/500

T_a = 15 + 1 + 2(1/r) = 18ms

Napomena: Kod kontinualne dodele samo jednom računamo vreme pozicioniranja, preostale staze, ukoliko ih ima, se čitaju bez ikakvog vremena pozicionaranja jer je glava već tu gde treba.

b) Sad moramo da računamo da pomeramo glavu svaki put. Već smo odredili da je rotaciono kašnjenje 1ms, to ostaje isto. Kao i vreme traženja. Ako mi celu traku pročitamo za 2ms, onda jedan sektor pročitamo za 0.25ms. (2ms / 8)

Sledi, T_a = (15 + 1 + 0.25) x 8 = 130ms

Drajver predstavlja SW komponentu za upravljanje UI uređajem, dok je kontroler HW komponenta. Kontroler ima svoje registre, memoriju i CPU specijalne namene. U njegove registre drajver smešta komandu.Nakon upisa svake komande drajver proverava da li kontroler uređaja prihvata tu komandu i da li je spreman da prihvati sledeću komandu.Nakon slanja svih komandi – Drajver se blokira i čeka da ga prekid od U/I uređaja deblokira (ako obrada traje) Nakon završetka operacije, drajver proverava da li je nastala greška

12. 2016 - MAR

UNIX file sistem definiše blokove veličine 2KB i adrese blokova dužine 4B. Svaki i-čvor sadrži 10 direktnih adresa blokova, jednu adresu na jednostruko indirektan i adresu na dvostuko indirektnе blokove

- a. Koja je najveća veličina datotekе?
- b. Razmotrite slučaj da je polovina datoteka veličine 1.5KB, a druga polovina 2KB. Koliko deo diska (u %) je neiskorišćen?
- c. Ukoliko bi promenili veličinu disk bloka na 1KB, da li bi poboljšali iskorišćenost diska i za koliko?
- d. Koliko disk operacija je neophodno da bi se pristupilo 700-om bloku datoteke radi modifikacije?

Zadnja

13. 2016 - JAN

a) Ukoliko se bitmara koja evidentira slobodne blokove na disku uništi, da li bi ste i kako mogli da joj obnovite za 1) FAT32 2) NTFS 3) Unix FS

b) Unix koristi indeksiranje pomoću i-čvora sa 10 direktnih blokova, blokove veličine 4KB i disk adrese od 4B i jednostrukе i dvostrukе indirektnе blokove. Koja je maksimalna veličina datoteke u ovom sistemu? Napisati pseudokod funkcije operativnog sistema za čitanje 3000-og bloka datoteke.

pod b je isto kao jan 2017 pod b i c samo se trazi 300 cvor umesto N-ti

14. 2015 - APR

Objasniš zašto je koriscenje bit vektora za evidentiranje slobodnih blokova pouzdanije i boljih performansi u odnosu na listu slobodnih blokova. UNIX file sistem je upravo rebusovan, te su bafceri i bavt datoteke /a/b/c/file, ukoliko file sistem koristi blokove od 4KB i 32-bitne adrese blokova i brojeve i-noda? Objasniš.

zadnja

15. 2015 - MAR

UNIX file sistem definiše blokove veličine 2KB i adrese blokova dužine 4B. Svaki i-čvor sadrži 10 direktnih adresa blokova, jednu adresu na jednostruko indirektan i adresu na dvostuko indirektnе blokove

- a. Koja je najveća veličina datotekе?
- b. Razmotrite slučaj da je polovina datoteka veličine 1.5KB, a druga polovina 2KB. Koliko deo diska (u %) je neiskorišćen?
- c. Ukoliko bi promenili veličinu disk bloka na 1KB, da li bi poboljšali iskorišćenost diska i za koliko?
- d. Koliko disk operacija je neophodno da bi se pristupilo 700-om bloku datoteke radi modifikacije?

zadnja

16. 2015 - JAN

a) Navesti i ukratko opisati osnovne metode za evidenciju slobodnog prostorana disku.

b) Windows 98 koristi lančanu listu blokova diska za svaku datoteku, koja se memoriše u FAT32, Windows 7 koristi NTFS, a Unix koristi indeksiranje pomoću i-čvora sa 12 direktnih blokova. Prepostavimo da sva tri OS-a koriste blokove diska veličine 2KB i adresu bloka od 4 B. Posmatrajmo datoteku veličine 2MB (tj. 1024 blokova) koja je smeštena u 16 nekontinualnih sekvenci od po 64 bloka. Napisati pseudokod funkcije operativnog sistema za modifikaciju i snimanje 500-og bloka datoteke u Windows 98, Windows 7 i Unix-u?

zadnja

V-peta

1. 2018 - OKT

Opisati osnovne karakteristike algoritama za planiranje diska FCFS, SSF, SCAN i C-SCAN. Disk ima 200 trasa, numerisanih 0 do 199 i u redu zahteva se nalaze zahtevi za trasama 86, 147, 91, 177, 94, 153, 102, 175, 132. Prepostavimo da je glava diska upravo završila pristup trasi 125 i da trenutno pristupa trasi 143. Prikažite sekvencu u kojoj će se zahtevi obradivati za svaku od sledećih politika raspoređivanja: FCFS, SSF, SCAN i C-SCAN

Bilo

2. 2018 - OKT

- a) Šta je direktorijum i koje informacije sadrži?
- b) Kako se pomoću direktorijuma povezuje ime datoteke sa sadržajem datoteke na disku? Dati primer kod Linux-a, Windows 7?

Zadnja

3. 2018 - JAN

- a) Koja je razlika između prekidima vodenog U/I (*Interrupt-driven I/O*) i DMA (*Direct memory access (DMA)*)?
- b) Koja je razlika između *block* orijentisanih i *stream* orijentisanih U/I uređaja i kako se to odlikuje u dizajnu OS?

Blok uređaji prenose podatke u blokovima fiksne velicine koji su adresibilni npr disk ili dvd.
Znakovni prihvataju i isporučuju tokove znakova(bajtova) koji nisu adresibilni npr stampaci,tastatura,miš

-Kako se to odlikuje u dizajnu???

Kod dma je direktni prenos podataka između uređaja i glavne memorije , dok je kod tehnike vodjene prekidima prenos između uređaja i glavne memorije omogućen preko procesora(slika sa 9. slajda)

4. 2017- OKT

Objasniti ukratko na koji način Virtual File System omogućuje da se u okviru operativnog Sistema pristupa različitim fajl sistemima.

Razmotriti datoteku sa 100 blokova i smatrati da je FCB (ili indeksni blok) već u memoriji. Izračunati broj U/I operacija neophodnih za kontinualnu, link-ovanu (ulančavanjem) ili indeksiranu dodelu blokova datoteke. U slučaju kontinualne dodele, pretpostaviti da nema prostora za širenje datoteke na početku, ali ima prostora na kraju datoteke. Sadržaj novog bloka je u memoriji.

- a) Blok je dodat na početku, b) sredini, c) kraju datoteke
- d) Blok je obrisan sa početka, e) sredine, f) kraja datoteke

5. 2017- APR

Šta je vektor prekida? Čemu služi? Gde je lociran? U kakvoj je relaciji sa drajverom uređaja i rutinom za obradu prekida (*interrupt handler*). Opisati njihovu povezanost i funkcionisanje na primeru U/I sistemskog poziva *receive* za čitanje podataka sa mrežne kartice i koji je redosled događaja za obavljanje ove U/I operacije i ko ih generiše. Šta se dešava sa procesom koji je pozvao *receive()* ?

Vektor prekida je adresa interrupt handler rutine, ili tabelu u fiksnom delu memorije gde su za sve prekide stavljene adrese interrupt handler rutina. Zatim mi na osnovu adrese interrupt handler rutine skočimo na adresu nju samu.

6. 2017- JAN

- a) Koja komponenta (sloj) U/I softvera vrši prevodenje logičke adrese zadate u obliku broja bloka u fizičku adresu cilindar/staza/sektor, a koja komponenta U/I softvera se izvršava kada kontroler diska prekidom označi kraj transfera.
b) Za sledeći niz zahteva 97, 129, 110, 186, 147, 35, 10, 84, 120 prikazati sekvencu u kojoj će se zahtevi obradivati za svaku od sledećih politika rasporedovanja: SSF, SCAN, F-SCAN (ako su posle 3-eg obradenog zahteva stigli novi zahtevi 130, 35, 80). Glava se nalazi na stazi 100, i kreće se prema nižim brojevima staza.

Slojevi

Sloj korisnickog nivoa

Sloj softvera nezavisnog od UI

Sloj softvera zavistan od UI-a

Rukovaoci prekida

HARDVER

//plavo je odg vec

Za prekide je posledji, a za prevodenje logicke u fiziku mora da bude onaj najbliži hardveru ali posto nema logike da ovaj za prekide radi i to onda su zavisni od UI-a(drajveri)

logicke->fizicku: Slojevi ui uređaja - da, praktično drajver to radi.

kontroler diska->Sloj za planiranje i upravljanje – može tako, to je interrupt handler, onaj najniži sloj.

SSF: 97, 84, 110, 120, 129, 130, 147, 186, 80, 35, 10

SCAN: od 100 pa dole(tako kaze zadatok) linija : 97,84,10 pa druga linija

35,80,110,120,129,130,147,186(u skripti je 35 posle 84 ali u zad pise da novi dolaze posle treceg obradenog-to valjda znaci da ne ulaze odmah u niz iz zadatka nego dok crtas)

FSCAN Imas dva reda zahteva prvi je onaj koji su dosli prvi a drugi red za dostigle zahteve. Prvo radis prvi red sa SCAN pa drugi red sa istim

97 84 35 10 110 120 129 147 186 130 80 35

NstepSCAN - podelis niz na n podsekvinci pa za svaku radis scan.Novi zahtevi koji dodju idu u novi niz.Za n=1 koristi se fifo./Ne trazi se ovde, nego da razjasnim sta je

7. 2016 - OKT2

Prikazati strukture podataka koje fajl sistem koristi kod sledećih šema dodele prostora na disku: (a) kontinualna, (b) ulančavanje blokova i (c) indeksiranje, d) indeksiranje u grupama blokova. Neka je fajl sistem X konstruisan tako da koristi blokove dužine 8 reči, pri čemu je svaka reč dužine 4 bajta. Disk koji se koristi za smeštanje fajl sistema ima 32 bloka. Inicijalni blok (blok 0) sadrži ulaz direktorijuma (adresara) koji se sastoji od imena fajla i pointera na prvi i-čvor u bloku 1. Struktura i-čvora je data na slici:

Reč	Vrednost
0	Pristupne dozvole
1	Veličina fajla u rečima
2	Direktni blok
3	Direktni blok
4	Direktni blok
5	Direktni blok
6	Jednostruki indirekt
7	Dvostruki indirekt

Fajl sistem X trenutno ima samo fajl PROBA dužine 16 reči podatka: prvi direktni indeks ukazuje na blok 31, a drugi na blok 29. Blokovi diska 4,7,10 i 15 su markirani kao neispravni. Slobodni blokovi se dodeljuju u redosledu 0,1,2,...,31. Operacija WRITE upisuje jedan blok. Ako fajlu PROBA treba dodati 100 reči koliko je WRITE operacija potrebno za ovo dodavanje? Prikazati stanje i-čvora, dodeljene i slobodne blokove pre i posle ove operacije dodavanja.

Zadnja

8. 2016 - OKT

Razmotriti sistem sa diskom sa 16 sektora po stazi i 512 B po sektoru. Disk se rotira brzinom od 7200rpm i ima prosečno vreme traženja (seek time) od 16ms. Posmatrati datoteku koja ima 24 bloka pri čemu 1 blok zauzima 1 sektor i izračunati ukupno vreme neophodno za pristup celoj datoteci ako se koriste metode alokacije a) kontinualna i b) ulančavanje pomoću tabele u memoriji (FAT) c) indeksiranje u grupama blokova (6 grupa po 4 kontinualna bloka).

Zadnja zbog ovog pod a,b,c

9. 2016 - SEP

Navesti i ukratko opisati osnovne metode za evidenciju slobodnog prostora na disku: Windows 98 koristi lančanu listu blokova diska za svaku datoteku, koja se memorise u FAT32, a UNIX koristi i-čvor. Pretpostavimo da obe OS-a koriste blokove diska veličine 4KB i adresu bloka od 4 B. Posmatrajmo datoteku veličine 2MB (tj. 512 blokova). Objasnite šta Windows 98, a šta UNIX treba da uradi da bi pročitao poslednji bajt iz ove datoteke, a šta da bi dodao novi blok (tj. 4KB novog sadržaja) na početak datoteke?

Zadnja ?

10. 2016 - JUN

Razmotriti sistem sa diskom sa 16 sektora po stazi i 512 B po sektoru. Disk se rotira brzinom od 5400rpm i ima prosečno vreme traženja (seek time) od 16ms. Posmatrati datoteku koja ima 22 bloka pri čemu 1 blok zauzima 1 sektor i izračunati ukupno vreme neophodno za pristup celoj datoteci ako se koriste metode alokacije a) ulančavanje pomoću tabele u memoriji (FAT) c) indeksiranje, c) kontinualne dodele, d) indeksiranjem u grupama blokova

Zadnja?

11. 2016 - APR

Prikazati strukture podataka koje fajl sistem koristi kod evidencije slobodnog prostora na disku. Neka je fajl sistem X konstruisan tako da koristi blokove dužine 8 reči, pri čemu je svaka reč dužine 4 bajta. Disk koji se koristi za smeštanje fajl sistema ima 32 bloka. Inicijalni blok (blok 0) sadrži ulaz direktorijuma (adresara) koji se sastoji od imena fajla i pointera na prvi i-čvor u bloku 1. Struktura i-čvora je data na slici:

Reč	Vrednost
0	Pristupne dozvole
1	Veličina fajla u rečima
2	Direktni blok <i>31</i>
3	Direktni blok <i>29</i>
4	Direktni blok
5	Direktni blok
6	Jednostruki indirektni
7	Dvostruki indirektni

Fajl sistem X trenutno ima samo fajl PET dužine 16 reči podataka: prvi direktni indeks ukazuje na blok 31, a drugi na blok 29. Blokovi diska 4, 7, 10, 15 su označeni kao neispravni. Slobodni blokovi se dodeljuju u redosledu 0, 1, 2, ..., 31. Operacija WRITE upisuje jedan blok. Ako fajlu PROBA treba dodati 100 reči koliko je WRITE operacija potrebno za ovo dodavanje? Prikazati stanje i-čvora, dodeljene i slobodne blokove pre i posle ove operacije dodavanja.

Fajl sistem X koristi:

1 blok = 8 reči

1 reč = 4B

Disk koji se koristi za smeštanje fajl sistema ima 32 bloka.

Inicijalni blok, blok 0, sadrži ulaz direktorijuma koji se sastoji od imena fajla i pointera na prvi i-čvor u bloku 1.

Struktura je data na slici.

Reč	Vrednost
0	Pristupne dozvole
1	Veličina fajla u rečima
2	Direktni blok
3	Direktni blok
4	Direktni blok
5	Direktni blok
6	Jednostruki indirektni
7	Dvostruki indirektni

Fajl sistem X ima trenutno samo fajl PET dužine 16 reči podataka: prvi direktni indeks ukazuje na blok 31, a drugi na blok 29. Blokovi diska 4, 7, 10, 15 su označeni kao neispravni. Slobodni blokovi se dodeljuju u redosledu 0, 1, 2, ..., 31. Operacija WRITE upisuje 1 blok. Ako fajlu PROBA treba dodati 100 reči koliko je WRITE operacija potrebno za ovo dodavanje? Prikazati stanje i-čvora, dodeljene i slobodne blokove pre i posle ove operacije dodavanja.

Ako se dodaje 100 RECI onda je to 13 blokova novih moramo da upišemo. Druga stvar koja je bitna je da nam treba jednostruki i dvostruki indirektni blok. Fajl PET ima samo

2 direktna bloka. Mi kad dodamo 13 nama treba 2 ova... Bitno je ovde da je disk adresa 1 reč. Ako je ovo 1 blok disk adresa je 1 reč. U jednom indirektnom bloku imamo 8 disk adresa. Nama treba 13, znači $2 + 8$ i treba nam dvostruki indirektni za još tri bloka, odnosno adresu. Znači samo treba staviti pravougaonik od 0 do 31 i markirati samo one koji su neispravni i alocirati. Treba nam 13 novih blokova i treba nam još 3 indirektna bloka. Jednostruki indirektni i 2 dvostruko indirektna. I samo se redom slažu brojevi i adrese.

12. 2016 - MAR

Šta je vektor prekida? Čemu služi? Gde je lociran? U kakvoj je relaciji sa drajverom uređaja i rutinom za obradu prekida (interrupt handler). Opisati njihovu povezanost i funkcionisanje na primeru U/I operacije *read* za čitanje bloka podataka sa diska i koji je redosled događaja za obavljanje ove U/I operacije i ko ih generiše.

Vektor prekida je adresa interrupt handler rutine, ili tabelu u fiksnom delu memorije gde su za sve prekide stavljenе adrese interrupt handler rutina. Zatim mi na osnovu adrese interrupt handler rutine skočimo na adresu nju samu.

???

13. 2016 - JAN

- a) Brzina okretanja diska je 7200 RPM. Vreme traženja je 9ms, a veličina disk sektora je 512 B, ulazi u 1024 sektora. Za koliko se vreme pročita sadržaj datoteke od 128 KB ako su blokovi datoteke, u veličine 4KB smesteni i sekvenčno, u) u četiri nezavisna run-a od po 8 blokova (NTFS), ili ako se koristi indeksiranje (Linux)?
- b) Za sledeći niz zahteva 27, 129, 110, 186, 147, 41, 10, 64, 120 prikazati sekvencu u kojoj će se zahtevi obradivati za svaku od sledećih politika rasporedovanja: SST, SCAN, F-SCAN (ako su posle 3-eg obradenog zahteva stigli zahtevi 131, 15, 20). Glava se nalazi na stazi 100, i kreće se prema nižim brojevima staza.

14. 2015 - APR

Pregostavite da imate disk sa rotacionom brzinom 15000 rpm, 512 B po sektoru, 400 sektora po stazi od 1MB koja je kontinualno snimljena na disk.

- a) Koliko je vreme prenosa ove datoteke sa diska u memoriju?
- b) Koliko je posećeno vreme pristupa za ovu datoteku?
- c) Koliko je rotaciono kretanje u ovom slučaju?
- d) Koliko je ukupno vreme za čitanje jednog sektora?
- e) Koliko je ukupno vreme za čitanje jedne staze?

15. 2015 - MAR

Šta je vektor prekida? Čemu služi? Gde je lociran? U kakvoj je relaciji sa drajverom uređaja i rutinom za obradu prekida (interrupt handler). Opisati njihovu povezanost i funkcionisanje na primeru U/I operacije *read* za čitanje bloka podataka sa diska i koji je redosled događaja za obavljanje ove U/I operacije i ko ih generiše.

16. 2015 - JAN

- a) Zašto je bilo implementirati najbolju strategiju za raspoređivanje (planiranje) pristupa disku?
 b) Za sledeći niz zahteva 26 37 100 14 88 33 99 64 67 12 prikazati sekvencu u kojoj će se zahtevi obradivati za svaku od sledećih politika rasporedivanja kao i broj kretanja glave: SSF, SCAN, C-SCAN, N-step-SCAN (N=3). Glava se nalazi na stazi 55, a prethodno je obradila zahtev na stazi 54.

a) Da bismo što je više moguće smanjili srednje vreme pozicioniranja glave diska (vreme traženja) i na taj način poboljšali performanse

b) zahtevi: 26,37,100,14,88,33,99,64,67,12
12,14,26,33,37,64,67,88,99,100 glava se nalazi na 55

SSF:64,67,88,99,100,37,33,26,14,12 (sve vece od 55 u rastuci redosled, pa sve manje od 55 u opadajuci)

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

F-SCAN: 64,67,88,99,100,37,33,26,14,12

N-stepScan(n=3) 26,37,100 14,88,33 99,64,67 12

N-Stepscan: 100, 37, 26, 14, 33, 88, 99, 67, 64, 12

Cscan:[bila na 54 pa 55 znaci ide ka vecim prvo]64,67,88,99,100,12,14,26,33,37