Drugi kolokvij iz predmeta *Operacijski sustavi* _____

30. lipnja 2023. Ime i prezime

Zadatke 1-8, 11 i 14 rješavati na ovim listovima.

1. (2) Na koje probleme treba paziti kod sinkronizacije dretvi?

   potpuni zastoj, izgladnjivanje

   mnogi studenti su ovo shvatili kao probleme algoritama za ostvarenje sinkronizacije, ili zašto treba dretve sinkronizirati

2. (2) Zašto je raspoređivanje podjelom vremena “bolje” od raspoređivanja po redu prispijeća?

   kraći poslovi prije dođu na red i budu posluženi

3. (2) Usporediti dinamičko upravljanje memorijom i straničenje. Navesti dobra i loša svojstva ova dva načina upravljanja.

upravljanje	prednosti	nedostatci
dinamičko	**jednostavan sklop, zaštita**	**fragmentacija, ne mogu se pokretati veliki programi**
straničenje	**zaštita, veliki programi, efikasno korištenje memorije**	**složen sklop, usporenje zbog promašaja**

4. (2) Navesti osnovne jedinice podataka sa jedinstvenom "adresom": u memoriji računala, na tvrdom disku (interno), u okviru datotečna sustava. Opisati "format" tih adresa, od čega se sastoje.

tip	osnovna jedinica podataka	format adrese (od čega se sastoji)
radna memorija	**oktet/bajt/B**	**broj**
tvrdi disk (HDD)	**sektor**	**površina, staza, sektor**
datotečni sustav	**blok (ili isto točno: datoteka)**	**redni broj bloka na particiji (za 2. rješenje: ime datoteke, put do datoteke kroz direktorije)**

5. (2) Zašto se koriste višediskovni sustavi (RAID)? Koje koristi donosi takav sustav naspram sustava kod kojeg diskovi nisu tako povezani?

   veći kapacitet
   efikasniji rad (brže čitanje/pisanje)
   zalihost - rade i kad se neki disk pokvari

6. (2) Navesti mehanizme komunikacije među dretvama unutar istog računalnog sustava. Koji je od njih najefikasniji ako dretve trebaju razmijenjivati velike količine podataka u jednom smjeru (npr. jedna dretva rezultate svoje obrade prosljeđuje drugoj na obradu)?

   zajednička memorija, poruke, cjevovod

   cjevovod za navedenu namjenu

7. (2) Koja su dva osnovna tip virtualizacije (na osobnom računalu, u poslužiteljskom centru)? Kratko opisati svojstva takvih virtualizacija.

   na osobnom računalu: virtualizacija na razini OS-a: postoji OS iznad kojeg ide virtualizacija, virtualno računalo i u njemu gostujući OS

   na poslužitelju: vritualizacija na razini sklopovlja: nema OS-a nego tanak sloj upravljača (hipervizor)

   drugi način je efikasniji jer su virtualna računala "bliže sklopovlju" ne prolaze kroz OS domaćina

8. (3) U nekom sustavu bez sklopa za prihvat prekida i bez programske potpore za prihvat prekida prema prioritetu treba posluživati tri vanjske jedinice. Prva generira prekid svakih 100 ms, druga svakih 200 ms, treća svakih 500 ms. Obrada svakog prekida traje 10 ms (uključujući prihvat PP i povratak iz prekida PiP). Koji su najgori a koji najbolji scenariji međusobne (ne)sinkronizirane pojave tih prekida, tj. koliko najduže traje čekanje od zahtjeva za prekid do početka obrade tog zahtjeva u jednom, tj. drugom slučaju?

   najgori slučaj je kad se u nekom trenutku zahtjevi poklope – tada će neki morati čekati kraj obrade prva dva, tj. bit će gotov tek za 30 ms
   npr. sva tri kreću u t=0

   u najboljim slučajevima se ne preklapaju obrade
   npr. P1 kreće prvi puta u t=0, P2 u t=20 ms i P3 u t=40 ms

9. (6) U nekom sustavu koji poslužuje korisnike obrada se obavlja u slijedećim koracima: (1) prihvati zahtjev; (2) dohvati nove podatke prema zahtjevu ili (3) dohvati stare podatke prema zahtjevu; (4) obradi dohvaćene podatke od (2) ili (3); (5) vrati rezultate korisniku.

   Programska potpora sastoji se od dretvi: `glavna(id)`, `dohvati_stare()`, `dohvati_nove()`. U sustavu može biti puno instanci tih dretvi (više dretvi `glavna`, ...). Glavna dretva obavlja operacije (1), (4) i (5) dok za (2) i (3) traži usluge ostalih dretvi. Nakon zaprimanja zahtjeva glavna dretva zahtjev prosljeđuje dretvama `dohvati_nove` ako to može u tom trenutku napraviti. U protivnom zahtjev prosljeđuje dretvama `dohvati_stare` (ili najprije čeka da to može napraviti). Dretve `dohvati_*` podatak dohvaćaju sa `dohvati(zahtjev)`. Nakon primitka odgovora glavna dretva obrađuje dobivene podatke te rezultat vraća korisniku. Nakon toga je glavna dretva spremna za prihvat novog zahtjeva.

   Komunikacija među dretvama obavlja se preko četiri zajedničke varijable: `zahtjev_novi`, `zahtjev_stari`, `odgovor_novi`, `odgovor_stari`. Svaka varijabla se sastoji od dva dijela: `id` – identifikacijskog broj zahtjeva (broja glavne dretve – svaka ima jedinstveni id) te `zahtjev` – samog zahtjeva (kazaljke na zahtjev). Strana koja želi poslati zahtjev to može napraviti tek kad je u toj varijabli `id` jednak nuli. Strana koja želi preuzeti može to uzeti tek kad `id` nije nula. Npr. glavna radi:

```
dretva glavna(id) {
   ponavljaj {
      zahtjev = čekaj_novi_zahtjev();
      ...
      ako je zahtjev_novi.id == 0 tada {
         zahtjev_novi = {id, zahtjev}
         ...
         dok je odgovor_novi.id != id
            //čekaj na signal da je odgovor postavljen
      } inače {
         //po potrebi pričekaj da možeš poslati zahtjev
         //pošalji zahtjev preko zahtjev_stari
         //čekaj na odgovor preko odgovor_stari
```

```
        }
        rezultat = obradi(odgovor)
        vrati rezultat klijentu
    }
}
```

Napisati pseudokod opisanih dretvi korištenjem monitora za sinkronizaciju. Operacije `dohvati()` i `obradi()` trebaju biti izvan monitora. Radno čekanje nije dozvoljeno.

```
dretva glavna(id) {
  ponavljaj {
    zahtjev = čekaj_novi_zahtjev();

    mutex_lock(m)
    ako je zahtjev_novi.id == 0 tada {
      zahtjev_novi = {id, zahtjev}
      cond_signal(novi_poslan)
      while odgovor_novi.id != id
        cond_wait(novi_odgovor, m)
      odgovor = odgovor_novi.odgovor
      odgovor_novi.id = 0
      cond_signal(novi_slobodan)
    } inače {
      while zahtjev_stari.id != 0
        cond_wait(stari_zahtjev, m)
      zahtjev_stari = {id, zahtjev}
      cond_signal(stari_poslan)
      while odgovor_stari.id != id
        cond_wait(stari_odgovor, m)
      odgovor = odgovor_stari
      odgovor_stari.id = 0
      cond_signal(stari_slobodan)
    }
    mutex_unlock(m)

    rezultat = obradi(odgovor)
    vrati rezultat klijentu
  }
}
```

```
dretva dohvati_nove() {
  ponavljaj {
    mutex_lock(m)
    cond_wait(novi_poslan, m)
    zahtjev = zahtjev_novi
    zahtjev_novi.id = 0
    mutex_unlock(m)

    rezultat = dohvati_novi(zahtjev)

    mutex_lock(m)
    while odgovor_novi.id != 0
      cond_wait(novi_slobodan, m)
    odgovor_novi = rezultat
    cond_broadcast(novi_odgovor)
    mutex_unlock(m)
  }
}

dretva dohvati_stare() {
  ponavljaj {
    mutex_lock(m)
    cond_wait(stari_poslan, m)
    zahtjev = zahtjev_stari
    zahtjev_stari.id = 0
    cond_signal(stari_zahtjev)
    mutex_unlock(m)

    rezultat = dohvati_stari(zahtjev)

    mutex_lock(m)
    while odgovor_stari.id != 0
      cond_wait(stari_slobodan, m)
    odgovor_stari = rezultat
    cond_broadcast(stari_odgovor)
    mutex_unlock(m)
  }
}
```

10. (4) U nekom sustavu nalaze se dretve A, B, C, D i E s prioritetima p(A)=1, p(B)=3, p(C)=2, p(D)=3 i p(E)=4. Sve se dretve raspoređuju prema SCHED_RR (prioritet pa podjela vremena) s kvantom vremena $T_q = 1$. U $t = 0$ sve su dretve blokirane (čekaju nešto). U $t = 1$ budi se dretva A, u $t = 2$ dretva B, u $t = 3$ dretva C, u $t = 4$ dretva D te u $t = 6$ dretva E. Svaka dretva treba četiri jedinice vremena za dovršetak svog posla (prije ponovna blokiranja). Pokazati izvođenje dretvi ako sustav ima:

a) (2) jedan procesor

b) (2) dva procesora

```
        A B B B D E E E E B D D D C C C C A A A
ili:    A B B D B E E E E D B D D C C C C A A A
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
      A B C D   E
```

b)

```
P1:  A A C D D D D C C C
P2:  - B B B B E E E E A A
  0 1 2 3 4 5 6 7 8 9 0 1 2
    A B C D   E
```

11. (3) U nekom sustavu koji koristi 32-bitovne adrese i veličinu stranice od 4 kB, tablica prevođenja procesa Px u nekom trenutku ima vrijednosti prema tablici 1. U svom radu proces generira adrese prema tablici 2. Kako će ih sklop za pretvorbu adresa prevesti, tj. što će napraviti?

Tablica 1. Tablica prevođenja

indeks okvira	zast. prisutnosti
0xABCD	1
0xDCBA	1
-	0
-	0
0xEF00	1

Tablica 2. Pretvorba adresa

logička adresa	fizička adresa (upisati)
0x0000	
0x1111	
0x2222	
0x4444	
0x8888	

Rješenje

logička adresa	fizička adresa
0x0000	**0xABCD000**
0x1111	**0xDCBA111**
0x2222	**promašaj**
0x4444	**0xEF00444**
0x8888	**prekid i zaustavljanje programa**

Mnogi nisu shvatili oznake 0xNEŠTO. Podsjetnik: prefiks 0x označava da je ovaj zapis broja u heksadekadskom brojevnom sustavu. 0x1234 == $1234_{(16)}$

Logička adresa rastavlja se na redni broj stranice i odmak. Obzirom da je veličina stranice 4 kB za odmak nam treba 12 najnižih bita, što su tri zadnje heksadekadske znamenke. Više znamenke (u ovom primjeru samo preostala jedna) je indeks stranice.

Tako se logička adresa: 0x4444 rastavlja na 0x4=4 kao indeks stranice te 0x444 kao odmak. Indeks stranice se preko tablice prevođenja zamijenjuje indeksom okvira (0x4 => 0xEF00) te konačna logička adresa je: 0xEF00444.

12. (4) Neki program zbraja matrice: `C = A + B` algoritmom:

```
za i=1 do N
  za j=1 do N
    C[j,i] = A[j,i] + B[j,i]
```

Matrice su u memoriji pohranjene po retcima (elementi `A[x,y]` i `A[x,y+1]` su susjedni).

Ako u jedan redak matrice stane u jednu stranicu, a program na raspolaganju ima tri okvira, koliko će promašaja izazvati program? Instrukcije programa i pomoćne varijable koriste druge stranice koje su u memoriji (nema promašaja zbog njihova korištenja). Algoritam zamjene stranica je FIFO.

za svako zbrajanje tri promašaja

ukupno: 3*N*N

13. (4) U NTFS datotečnom sustavu neka datoteka je pohranjena u tri dijela: prvih 10 MB kompaktno smještenih na disku počevši od bloka 10001, drugih 5 MB kompaktno smještena na disku počevši od bloka 100001 te zadnjih 100 MB potpuno fragmentirano. Veličina bloka datotečna sustava je 16 kB.
    a) (2) Napisati tablicu s opisom smještaja (barem dio koji je zadan) te navesti ukupni broj redaka te tablice.
    b) (2) U kojem bloku na disku se nalazi 777. blok datoteke?

    a)
    10MB: 10 MB / 16 KB = 10*1024/16 = 10*64 = 640 blokova
    5 MB: 320 blokova
    100 MB: 6400 blokova

    ```
    VCN – LCN    – #
    ------------------
    1   – 10001  – 640
    641 – 100001 – 320
    + 6400 redaka za fragmentirani dio = ukupno 6402 retka
    ```

    b)
    777. blok datoteke => u drugom dijelu
    777 – 640 = 137
    100001 + 136 = 100137

14. (2) Za ostvarenje pohrane u nekom sustavu na raspolaganju stoji 20 diskova, svaki kapaciteta 8 TB. Sustavu je potrebno (dovoljno) 50 TB diskovna prostora.
    a) (1) Ako su potrebne maksimalne performanse i dovoljna je zaštita od kvara jednog diska, koji RAID treba odabrati (0, 1, 5 ili 6) i koliko diskova koristiti?
    b) (1) Ako je potrebna maksimalna pouzdanost koji RAID treba odabrati (0, 1, 5 ili 6) i koliko diskova koristiti?

    a) RAID 5, svi diskovi radi performansi (paralelno čitanje/pisanje)

    b) RAID 6, minimalan broj diskova (manja vjerojatnost kvara): (N-2)*8 >= 50 => devet diskova