# Drugi kolokvij iz predmeta *Operacijski sustavi*

28. lipnja 2017.

PISATI ČITKO $\Longrightarrow$ NEČITKO $\equiv$ KRIVO

1. (2) Čemu služi Bankarev algoritam (u okviru operacijskih sustava)? Što on može pokazati?

   **Rješenje**

   Bankarev algoritam služi za provjeru stanja u sustavu u kojem ima više dretvi i više sredstava koja su njima potrebna. Stanje može biti stabilno – ima načina da sve dretve završe s radom bez da se dogodi potpuni zastoj, ili nestabilno (potpuni zastoj će se neizbježno dogoditi).

2. (2) Opisati način raspoređivanja dretvi prema SCHED_FIFO.

   **Rješenje**

   - dretve se raspoređuju prema prioritetu
   - kada ima više dretvi ista prioriteta, drugi kriterij je red prispijeća

3. (2) Kad proces zatraži otvaranje datoteke, koje sve radnje operacijski sustav treba napraviti (koje elemente treba dohvatiti s diska, čime se oni proširuju prije rada s tom datotekom, ...)?

   **Rješenje**

   - dohvatiti opisnik datoteke iz datotečne tablice (napraviti kopiju u radnom spremniku)
   - proširiti opisnik kazaljkom (na trenutni položaj u datoteci)
   - pripremiti međuspremnike za rad s datotekom
   - vratiti redni broj opisnika procesu (redni broj brojeći samo opisnike datoteka otvorenih u procesu)

4. (2) Kratko opisati dva osnovna načina virtualizacije. Alate VMware Player, VMware ESX/ESXi, Xen Project, Oracle VM Server, VirtualBox svrstati u odgovarajuće načine.

   **Rješenje**

   1. virtualizacija korištenjem operacijskog sustava domaćina
   2. virtualizacija na razini sklopovlja (korištenjem hipervizora)

   Prvi radi tako da se okruženje za virtualna računala (gosta) stvara unutar postojećeg operacijskog sustava (domaćina).
   Drugi radi tako da se postavi program (hipervizor) odmah iznad sklopovlja, koji predstavalja tanak sloj iznad tog sklopovlja – virtualizira korištenje tog sklopovlja, iznad njega idu operacijski sustavi (više njih, ravnopravno ili kako im hiprevizir da)

   U 1: VMware Player, VirtualBox
   U 2: VMware ESX/ESXi, Xen Project, Oracle VM Server

5. (4) U nekom sustavu sa sklopom za prihvat prekida javljaju se prekidi: $P_1$ svakih 50 ms, $P_2$ svakih 200 ms te $P_3$ svakih 500 ms. Prioritet prekida određen je brojem ($P_3$ ima najveći prioritet). Procedura za prihvat prekida (PP) traje 0,5 ms a procedura za povratak iz prekida (PiP) 0,5 ms. Obrada svakog prekida traje po 4 ms. Koji dio procesorskog vremena tog sustava (u postocima) se potroši na obrade navedenih prekida?

   **Rješenje**

   po jednoj obradi: PP+obrada+PIP = 0,5+4+0,5 = 5 ms
   dio procesorskog vremena koji P1 uzima: 5/50

```
dio procesorskog vremena koji P2 uzima: 5/200
dio procesorskog vremena koji P3 uzima: 5/500
Ukupno: 5/50+5/200+5/500 = 0,1+0,025+0,01 = 0,135 = 13,5 %
(Priznata su i rješenja u koja nisu uračunati kućanski poslovi)
```

6. (4) Kada neki kupac dobavi sve željene proizvode u svoju košaricu dolazi do blagajni. U prodavaonici postoje dvije blagajne i zaseban red za svaku. Kupac će stati u kraći red (ili prvi red, ako su oba jednako duga). Ukoliko se sustav simulira dretvama i pseudokod blagajnica je kao u nastavku (i ne može se mijenjati!), pokazati pseudokod kupaca s aktivnostima: "napuni košaricu", "stavi proizvode na traku" (tek kad ga blagajnica pozove), "plati račun", "uzmi proizvode i napusti prodavaonicu".

```
dretva blagajnica(i) // i je 1 za prvu, 2 za drugu dretvu blagajnice
{
    ponavljaj {
        PostaviOSEM(S0[i]) //poziva idućeg kupca
        ČekajOSEM(S1[i])   //čeka da idući kupac sve stavi na traku
        očitaj cijene s proizvoda
        izradi i daj račun
        PostaviOSEM(S2[i]) //signalizira kupcu da plati navedeni iznos
        ČekajOSEM(S3[i])   //čeka da se novci stave na pult
        spremi novce i vrati ostatak
        PostaviOSEM(S4[i]) //kupac može otići sa svojim proizvodima
    }
    do kraja radna vremena
}
```

Rad s dužinom redova raditi u kritičnom odsječku. Svi redovi semafora uređeni su prema redu prispijeća (FIFO). Navesti početne vrijednosti svih korištenih varijabli i semafora (početne vrijednosti semafora navedenih u gornjem kodu su nule).

**Rješenje**

```
red[2] = {0,0}
dretva kupac()
{
    napuni košaricu
    ČekajBSEM(KO)
    ako je red[1] <= red[2] tada
        i = 1
    inače
        i = 2
    red[i] = red[i] + 1
    PostaviBSEM(KO)

    ČekajOSEM(S0[i])
    stavi proizvode na traku
    PostaviOSEM(S1[i])
    ČekajSEM(S2[i])
    plati račun
    PostaviOSEM(S3[i])
    ČekajOSEM(S4[i])
    ČekajBSEM(KO)
    red[i] = red[i] - 1
    PostaviBSEM(KO)
    uzmi proizvode i napusti prodavaonicu
}
```

7. (4) X procesa proizvođača komunicira s Y procesa potrošača preko zajedničkog međuspremnika M u koji stane N poruka. Simulirati sustav dretvama (ostvariti funkcije za dretve `proizvođač()` i `potrošač()`). Za sinkronizaciju koristiti monitore (i dodatne varijable). Sve su dretve cikličke. U svakoj iteraciji svaki proizvođač stavlja po jednu poruku na prvo prazno mjesto, dok potrošači uvijek uzimaju prvu (najstariju) nepročitanu poruku. Stvaranje nove poruke (kod proizvođača) simulirati s `p = dohvati()`, a obradu poruke kod potrošača sa `obradi(p)`. Navedene operacije moraju biti izvan dijela koji se sinkronizira (razne dretve trebaju moći te operacije raditi paralelno, nad različitim porukama).

**Rješenje**

```
m - monitor
čekaj_poruku - red za potrošače
čekaj_mjesto - red za proizvođače
M[N] - međuspremni
ULAZ - prvo prazno mjesto (tako di treba staviti novu poruku)
IZLAZ - prvo puno mjesto (od kuda treba uzeti poruku)
PUNIH - koliko ima nepročitanih poruka u M

proizvođač () {
    ponavljaj {
        p = dohvati()
        Uđi_u_monitor(m)
        dok je PUNIH == N radi
            Čekaj_u_redu_uvjeta ( čekaj_mjesto, m )
        PUNIH++
        M[ULAZ] = p
        ULAZ = (ULAZ + 1) MOD N
        ako je ( PUNIH == 1 ) //može i bez provjere
            Oslobodi_iz_reda_uvjeta( čekaj_poruku, m )
        Izađi_iz_monitora(m)
    }
    zauvijek
}

potrošač () {
    ponavljaj {
        Uđi_u_monitor(m)
        dok je PUNIH == 0 radi
            Čekaj_u_redu_uvjeta ( čekaj_poruku, m )
        PUNIH--
        r = M[IZLAZ]
        IZLAZ = (IZLAZ + 1) MOD N
        ako je ( PUNIH == N-1 ) //može i bez provjere
            Oslobodi_iz_reda_uvjeta( čekaj_mjesto, m )
        Izađi_iz_monitora(m)
        obradi(r)
    }
    zauvijek
}
```

8. (4) Nekom poslužitelju dolaze tri tipa posla. Prvi poslovi dolaze prosječno 40 u sekundi, drugi 60 u sekundi te trećih 10 u sekundi (prema Poissonovoj razdiobi). Prosječno trajanje obrade prvih traje 10 ms, drugih 5 ms te trećih 10 ms (prema eksponencijalnoj razdiobi). Zbog velikog opterećenja odlučeno je da se nabave još dva poslužitelja istih svojstava te da svaki poslužitelj obrađuje svoj tip posla. Koliko je opterećenje početna poslužitelja (uz sva tri tipa poslova), te

koliko će biti opterećenje tih poslužitelja nakon preseljenja?

**Rješenje**

$\alpha_1 = 40\ s^{-1}, \alpha_2 = 60\ s^{-1}, \alpha_3 = 10\ s^{-1}$
$1/\beta_1 = 10\ ms = 0,01\ s, 1/\beta_2 = 5\ ms = 0,005\ s, 1/\beta_3 = 10\ ms = 0,01\ s$
$\rho_0 = \alpha_1/\beta_1 + \alpha_2/\beta_2 + \alpha_2/\beta_2 = 40 \cdot 0,01 + 60 \cdot 0,005 + 10 \cdot 0,01 = 0,4 + 0,3 + 0,1 = 0,8$
$\rho_1 = \alpha_1/\beta_1 = 0,4$
$\rho_2 = \alpha_2/\beta_2 = 0,3$
$\rho_3 = \alpha_3/\beta_3 = 0,1$

9. (4) U nekom jednoprocesorskom sustavu javljaju se četri zadatka: P1 u 10., P2 u 20., P3 u 30., i P4 u 40. jedinici vremena. Zadaci se raspoređuju prema prioritetu. Prioritet zadatka jednak je broju (P4 ima najveći prioritet). P1 treba 10 MB, P2 20, P3 30 te P4 30 MB. Svaki zadatak treba 30 jedinica vremena za dovršetak svog posla. Pohrana jednog procesa (neovisno o veličini) iz radnog u pomoćni spremnik neka traje 5 jedinica vremena. Isto toliko traje i učitavanje jednog procesa s pomoćnog u radni spremnik. Pretpostaviti da sustav ne paralelizira učitavanje/spremanje procesa (najviše jedan proces se učitava s pomoćnog spremnika ili sprema na njega). Procesi su već pripremljeni na pomoćnom spremniku. Pokazati rad sustava do završetaka svih zadataka, ako se koristi dinamičko upravljanje spremnikom veličine 60 MB.

**Rješenje**

| t | 123456 MEM | Akt | Pripr | Pripr-disk | |
|---|---|---|---|---|---|
| 0 | ------ | - | - | - | |
| 5 | ------ | - | - | - | |
| 10 | *----- | - | - | P1 | +P1, P1 se učitava |
| 15 | 1----- | P1 | - | - | P1 radi |
| 20 | 1**--- | P1 | - | P2 | +P2, P2 se učitava |
| 25 | 122--- | P2 | P1 | | |
| 30 | 122*** | P2 | P1 | P3 | +P3, P3 se učitava |
| 35 | 122333 | P3 | P2,P1 | | P3 radi |
| 40 | %22333 | P3 | P2 | P4, P1 | +P4, P1 natrag na disk |
| 45 | %%%333 | P3 | - | P4, P2, P1 | P2 natrag na disk |
| 50 | ***333 | P3 | - | P4, P2, P1 | P4 se učitava |
| 55 | 444333 | P4 | P3 | P2, P1 | P4 radi |
| 60 | 444333 | P4 | P3 | P2, P1 | P4 radi |
| 65 | 444333 | P4 | P3 | P2, P1 | P4 radi |
| 70 | 444333 | P4 | P3 | P2, P1 | P4 radi |
| 75 | 444333 | P4 | P3 | P2, P1 | P4 radi |
| 80 | 444333 | P4 | P3 | P2, P1 | P4 radi |
| 85 | **-333 | P3 | - | P2, P1 | -P4, P3 radi, P2 se učitava |
| 90 | 22*333 | P3 | P2 | P1 | P3 radi, P1 se učitava |
| 95 | 221--- | P2 | P1 | - | -P3, P2 radi |
| 100 | 221--- | P2 | P1 | - | P2 radi |
| 105 | 221--- | P2 | P1 | - | P2 radi |
| 110 | 221--- | P2 | P1 | - | P2 radi |
| 115 | --1--- | P1 | - | - | -P2, P1 radi |
| 120 | --1--- | P1 | - | - | P1 radi |
| 125 | --1--- | P1 | - | - | P1 radi |
| 130 | --1--- | P1 | - | - | P1 radi |
| 135 | ------ | - | - | - | -P1 |

10. (4) U sustavu s virtualnim spremnikom, veličina okvira je N=4 riječi, a okviri se pune na zahtjev. Algoritam zamjene stranica je LRU. Matrica $A[4,4]$ je pohranjena po retcima (na susjednim lokacijama se mijenja desni indeks). Koliko promašaja će izazvati prikazani program ako za matricu A u radnom spremniku postoje dva okvira? Zanemariti promašaje zbog dohvata instrukcija samog programa i pristupa pomoćnim varijablama.

```
t = 0;
za i=1 do 3 {
    za j=i+1 do 4 {
        t = t + A[i,j];
        t = t * A[j,i];
    }
}
```

**Rješenje**

```
zahtjevi: A[1,2], A[2,1], A[1,3], A[3,1], A[1,4], A[4,1],
          A[2,3], A[3,2], A[2,4], A[4,2],
          A[3,4], A[4,2]
svaki red => jedna stranica;
zahtjevi prema stranicama (prvi indeks u A):
          1 2 1 3 1 4 2 3 2 4 3 4
LRU s dva okvira:
   1  2  1  3  1  4  2  3  2  4  3  4
- [1] 1 (1) 1 (1) 1 [2] 2 (2) 2 [3]
-  - [2] 2 [3] 3 [4] 4 [3] 3 [4] 4 (4)
[] => promašaji
() => pogotci
Ukupno promašaja: 8
```

11. (3) Opis smještaja neke datoteke u NTFS datotečnom sustavu zadan je tablicom:

| VCN | LCN | # |
|---|---|---|
| 1 | 1000 | 50 |
| 51 | 2000 | 100 |

Ukoliko je veličina bloka 4 kB, veličina sektora 512 B (u jednom je bloku 8 uzastopnih sektora), odrediti sektor na disku u kojem se nalazi 1. bajt 70. bloka datoteke. Pretpostaviti da prvi blok diska zauzima sektore 1-8, drugi 9-16 i tako dalje redom.

**Rješenje**

```
prvih 50 blokova datoteke je na disku u blokovima 1000-1049
idućih 100 blokova datoteke je na disku u blokovima 2000-2099
70. blok datoteke => 2000+19 = 2019. blok na disku
1. bajt 2019. bloka je u prvom sektoru tog bloka (od njih 8)
prije njega ima 2018 blokova s 8 sektora:
2018 * 8 + 1= 16144 + 1 = 16145. sektor
```

12. (2) U nekom sustavu koji koristi RAID 0 nalazi se 6 jednakih diskova. Za diskove je poznat $MTTF = 15$ godina. Koliko je očekivano vrijeme rada sustava do gubitka podataka?

**Rješenje**

```
Dovoljno je da se bilo koji disk pokvari pa da dođe do gubitka
podataka.
Stoga je formula:
```

```
MTTF_S = MTTF/N = 15/6 = 2,5 godina
```

13. (3) Pokazati razmijene poruke među čvorovima $P_1$, $P_2$ i $P_3$ ukoliko čvor $P_2$ zatraži ulazak u kritični odsječak korištenjem algoritma Ricart-Agrawala. Neka su početne vrijednosti lokalnih satova: $C_1 = 237$, $C_2 = 253$ te $C_3 = 371$. Niti jedan čvor trenutno nije u kritičnom odsječku niti je dao zahtjev za ulazak (prije zahtjeva od $P_2$).

**Rješenje**

```
prema algoritmu:
- čvor P2 povećava svoj lokalni sat C2 = 254
- šalje poruku Zahtjev(2,254) prema čvorovima P1 i P3
- P1 prima poruku:
  * ažurira svoj sat C1 = max{237, 254} + 1 = 255
  * obzirom da P1 ne traži ulaz, odmah šalje odgovor(2)
- isto i P3 (C3=372)
- P2 prima oba odgovora te ulazi u KO
- po završetku, obzirom da kod P2 nitko nema zahtjev, nema slanja
novih poruka
```