Operacijski sustavi, 2. kolokvij, 1. 7. 2021. Student _____

1. U nekom sustavu koji nema sklop za prihvat prekida niti programsko rješenje za prihvat i obradu prekida prema prioritetu, postoje tri izvora prekida. Prvi izvor P1 generira zahtjeve za prekid svake sekunde i treba 30 ms za obradu (uključujući i kućanske poslove: spremanje konteksta, određivanje uzroka prekida, …). Drugi P2 generira zahtjeve svakih 200 ms i treba 5 ms za obradu. Treći P3 se javlja svakih 50 ms i treba 1 ms za obradu. P3 ima najveći prioritet, slijedi P2 te P1 s najmanjim (prioriteti se gledaju samo u postupku prihvata prekida, kad se traži uzrok prekida). Pojave navedenih izvora prekidi ne moraju biti sinkronizirane (npr. da svi kreću s odbrojavanjem u 0. sekundi; primjerice P1 može se javljati u 0,372; 1,372; 2,372… P2 u 0,117; 0,317 itd.).

   a. (1) U najgorem slučaju, koliko će proteći od zahtjeva P3 do početka njegove obrade?

   P3 ima najveći prioritet pa ako se pojavi istovremeno s drugima prvi će se detektirati – to nije problem. Problem jest ako on dođe odmah iza nekog drugog prekida koji se već prihvatio – onda će morati čekati kraj te obrade. Obrada od P1 traje 30 ms, obrada od P2 5 ms.

   Znači, najgori slučaj je kad P3 dođe nakon P1 i tada mora čekati 30 ms

   b. (3) Ako se u nekom trenutku pokrene program P koji treba 1000 ms procesorskog vremena, koliko će u najgorem slučaju taj proces biti u sustavu (od pokretanja do završetka)?

   U najgorem slučaju P se javlja istovremeno sa sva tri prekida: onda i na početku mora čekati njihovu obradu. Također, u svom radu P je opet prekidan svim tim prekidima. Očito će stoga trajati više od 1000 ms.
   U prvoj sekundi rada programa P (1000 ms) na prekide će se potrošiti:
   (1000/1000)*30 ms + (1000/200)*5 + (1000/50)*1 = 30 + 25 + 20 ms = 75 ms
   Tih 75 ms će program morati raditi u slijedećoj sekundi, ali opet uzimajući u obzir prekide:
   30 ms (P1) + 5 ms (P2) + 1 ms (P3), pa onda za P do 50 ms ima 50-36=14 ms, ostaje 75-14=61
   U 50. ms 1 ms (P3) te do 100. ms za P ostaje 49 te još ostaje 61-49 = 12
   U 100. ms 1 ms (P3) te do 150. ms ostaje 50 ms za P, ali on treba samo 12, tj. bit će gotov u 113. ms, tj. ukupno će se u sustavu zadržati 1113 ms.

2. (4) U neko računalo postavljeno je 20 diskova, svaki kapaciteta 2 TB. (*rješavati na ovom papiru*)

   a. Koliki bi korisni kapacitet bio kad bi se koristili:

   RAID-0 **40 TB** RAID-1 **20 TB** RAID-5 **38 TB** RAID-6 **36 TB**

   b. Ako je dovoljno 25 TB korisna kapaciteta, koji RAID odabrati i koliko diskova treba tada koristiti a da bi sustav bio najotporniji (najdulje radio i u slučaju kvarova i popravaka dok radi)?

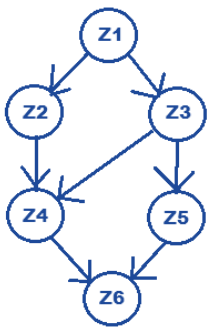
   Za 25 TB trebaju 13*2+2 diska = 15 diskova

   Pet diskova se neće koristiti.

   Zašto ne sve staviti? Veći broj diskove = veća vjerojatnost da će se neki od njih pokvariti.

3. (4) Neki posao podijeljen je u šest zadataka Z_1 ... Z_6. Poznato je da će rezultat biti ispravan ako se zadaci izvode slijedno: $Z_1 > Z_2 > Z_3 > Z_4 > Z_5 > Z_6$. Domene i kodomene zadataka su sljedeće: Z_1:{D=M_1; K=M_2}, Z_2:{D=M_2; K=M_3}, Z_3:{D=M_4; K=M_1}, Z_4:{D=M_3; K=M_4}, Z_5:{K=M_1}, Z_6:{D=M_1; K=M_4}. Odrediti maksimalno paralelni sustav zadataka (i prikazati ga usmjerenim grafom) uzimajući u obzir njihove domene i kodomene te međusobni odnos u lancu.

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



4. (4) Stanje nekog sustava u promatranom trenutku je sljedeće: dretva D_3 je aktivna, D_1 je u redu pripravnih, D_4 u redu semafora OSEM[1], te D_2 i D_5 u redu odgođenih: D_5 treba čekati još jedan otkucaj sata, a D_2 3 (ukupno). Red pripravnih dretvi je uređen prema prioritetu. Prioritet dretve određen je njenim indeksom – dretve s većim indeksom imaju veći prioritet (D_5 ima najveći). U tom stanju dogodi se sljedeći niz poziva jezgrinih funkcija (svaki idući se događa neko kratko vrijeme nakon što je prethodni bio gotov): 1. `Otkucaj_sata();` 2. `Započni_UI(1);` 3. `Postavi_OSEM(1);` 4. `Odgodi(5)`. Prikazati stanje sustava nakon svakog od tih poziva.

	početno stanje	nakon `Otkucaj_sata()`	nakon `Započni_UI(1)`	nakon `Postavi_OSEM(1)`	nakon `Odgodi(5)`
`Aktivna_D`	3	**5**	**3**	**4**	**3**
`Pripravne_D`	1	**3 1**	**1**	**3 1**	**1**
`OSEM[1]`	4	**4**	**4**	**- (.v=0)**	**- (.v=0)**
`Odgođene_D`	5^1 2^2	**2^2**	**2^2**	**2^2**	**2^2 4^3**
`UI[1]`	--	**--**	**5**	**5**	**5**

5. (4) Neki sustav koristi raspoređivanje prema prioritetu kao osnovnom kriteriju te prema podjeli vremena kao dodatnom koji se koristi kad prvi kriterij ne daje samo jednu dretvu. U nekom trenutku u sustavu se nalaze dretve A, B, C, D, E. Odnos prioriteta dretvi je: prio(A) < prio(B) = prio(D) < prio(C) = prio(E). Dretva A treba još 30 ms za završetak, dretva B 50 ms, dretva C 20 ms te dretve D i E po 30 ms. Prikazati stanje procesora od trenutka t_0 pa dok sve dretve ne završe s radom ako je kvant vremena T_q=10 ms.

C	**E**	**C**	**E**	**E**	**B**	**D**	**B**	**D**	**B**	**D**	**B**	**B**	**A**	**A**	**A**				

6. (4) U **8-procesorskom** sustavu s koji raspolaže s 22 okvira za programe pojavljuju se zahtjevi za pokretanje procesa (svaki sa samo jednom dretvom) i to redom: u t=5 ms P1 koji zauzima 10 stranica, u t=10 ms P2 koji zauzima 5 stranica, t=15 ms P3 koji zauzima 8 stranica, u t=20 ms P4 koji zauzima 6 stranica te u t=25 ms P5 koji zauzima 8 stranica. Svi su procesi već pripremljeni na pomoćnom spremniku (disku). Zahtjevi se poslužuju na sljedeći način. Svakom procesu koji se izvodi treba osigurati minimalno 5 okvira – ako to nije moguće proces čeka. Ako ima slobodnih okvira procesu se dodijeli i više od 5 okvira, ali ne više od 8. Ako je nekom procesu dodijeljeno više od 5 okvira, a neki drugi zahtjev se ne može pokrenuti (jer nema 5 slobodna), onda se okviri oduzimaju (jedan po jedan) procesima koji imaju najviše dodijeljenih okvira, do zadovoljenja uvjeta (ili dok svi imaju minimalno 5). Prikazati stanje sustava (okvire i tko ih koristi) do posluživanja svih procesa. Svaki proces treba 30 ms procesorskog vremena (jednog procesora).

t=0	**P1 (8)**																					
t=10	**P1 (8)**								**P2(5)**													
t=15	**P1 (8)**								**P2(5)**					**P3(8)**								
t=20	**P1 (6)**						**P4(2)**		**P2(5)**					**P3(6)**						**P4(3)**		
t=25	**P1(5)**						**P4(2)**		**P2(5)**					**P3(5)**						**P4(3)**		
t=30	**P5(6)**						**P4(2)**		**P2(5)**					**P3(5)**					**P5**	**P4(3)**		
t=40	**P5(6)**						**P4(2)**							**P3(5)**					**P5**	**P4(3)**		
t=45	**P5(6)**						**P4(2)**												**P5**	**P4(3)**		
t=50	**P5(6)**																		**P5**			
t=60																						

7. (4) Neki sustav upravljanja treba ostvariti korištenjem tri dretve `dohvati(id)` te jednom dretvom `obradi()`. Svaka dretva `dohvati(id)` provjerava svoj dio sustava funkcijom `provjera(id)`, gdje `id` predstavlja identifikacijski broj dretve. Kada funkcija `provjera(id)` vrati nulu tada taj dio sustava ne treba dodatnu pažnju te dotična dretva `dohvati` spava 5 sekundi prije ponovne provjere. U protivnom, kada je povratna vrijednost različita od nule, dretva `dohvati` treba dretvi `obradi` dostaviti svoj `id` i čekati da dretva `obradi` riješi taj problem prije nego li dretva `dohvati` ponavlja provjeru. Dretva `obradi` čeka da joj netko dojavi potrebu za obradom. Kad joj se dojavi ona događaj obrađuje funkcijom `obradi_događaj(id)`. Napisati pseudokod dretvi korištenjem monitora za sinkronizaciju. Akcijama koje su nezavisne dopustiti paralelni rad (`provjera` i `obradi_događaj` različitih dretvi, tj. `id`-jeva). Napisati početne vrijednosti korištenih varijabli.

```
m – monitor
čekaj_red – red uvjeta
čekaj_obradu – red uvjeta
čekaj_zahtjev – red uvjeta
z = 0 – zajednička varijabla; 0 kad nema zahtjeva, id zahtjeva kad ima

dretva dohvati(id) {
   ponavljaj {
      status = provjera(id)
      ako je (status == 0) {
         spavaj(5 sekundi)
      }
      inače {
         Uđi_u_monitor(m)
         dok je (z != 0)
            Čekaj_u_redu_uvjeta(čekaj_red, m)
         z = id
         Oslobodi_iz_reda_uvjeta(čekaj_zahtjev)
         Čekaj_u_redu_uvjeta(čekaj_obradu, m)
         Izađi_iz_monitora(m)
      }
   }
}
dretva obradi() {
   ponavljaj {
      Uđi_u_monitor(m)
      dok je (z == 0)
         Čekaj_u_redu_uvjeta(čekaj_zahtjev, m)
      obradi_događaj(z)
      z = 0
      Oslobodi_iz_reda_uvjeta(čekaj_obradu)
      Oslobodi_iz_reda_uvjeta(čekaj_red)
      Izađi_iz_monitora(m)
   }
}
```

8. (4) Koliko će promašaja izazvati prikazani algoritam, ako za matricu A u radnom spremniku postoje tri okvira? Veličina okvira jest N riječi, a algoritam zamjene stranica je LRU. Zanemariti promašaje zbog dohvata instrukcija i pomoćnih varijabli (gledati samo pristup elementima matrice).

```
za i = 1 do N
  za j = 1 do N
     A[i+1,j] = A[i,j] + A[i+1,j] //dohvati A[i,j], dohvati A[i+1,j], zbroji, spremi
```

```
Zahtjevi idu redom:
   1 2 2 1 2 2 … 1 2 2 | 2 3 3 …
- (1)1 | | | | … | | | | | 1 | …
-  -(2)| | | | … | | | | | 2 | …
-  - - | | | | … | | | | |(3)|
S tri okvira samo su prva dva promašaja za i=1, još jedan za i=2 itd
Tj. ukupno je N promašaja!
```

9. Neki disk ima 6 površina na kojima su zapisani podaci. Na svakoj stazi nalazi se 2000 sektora, svaki sektor sadrži 512 B. Disk se okreće sa 6000 okr/min. Pri čitanju disk čita cijelu stazu u interni spremnik te prenosi potrebne sektore u radni spremnik brzinom 4,096 Gbita/s (ne 4096; G = 10^9). Za vrijeme prijenosa ne može se čitati s diska, ali može se pomicati glavu. Datoteka veličine 50 MB kompaktno je smještena na disku (MB = 2^{20} B).

   a. (2) Kako su smješteni podaci te datoteke na disku (koliko punih/djelomičnih staza, cilindara, …)?

   veličina staze = 512 B * 2000 = 1000 KB

   50 MB / 1000 KB = 50*1024/1000 = 51,2 staze => 51 pune staze + 0,2 52. staze (0,2*2000=400 sektora)

   52/6=8 i nešto = 8 punih cilindara + na 9. cilindru: 3 pune staze i 400 sektora na 4. stazi

   b. (2) Koliko traje čitanje te datoteke u radni spremnik, ako je vrijeme traženja staze T_{seek} = 5 ms te vrijeme postavljanja na susjednu stazu T_1 = 0,5 ms.

   $t = T_{seek} + [(T_R/2+T_R+T_P)*5+(T_R/2+T_R+\max(T_P,T_1))]*8 + [(T_R/2+T_R+T_P)*3+(T_R/2+T_R+T_P*0,2)]$

   $T_R = 1/\omega$ = 1/(6000 okr/min) = 1/(6000/60 okr/s) = 1/100 s = 10 ms => $T_R/2$ = 5 ms

   T_P = vel. staze / brzina prijenosa = 1000 KB / 4,096 Gbita/s = 1000*1024*8 bita / (4,096 * 10^9 bita/s) =

   $T_P = 2 * 10^{-3}$ s = 2 ms => $\max(T_P, T_1) = T_P$ => $t = T_{seek} + (T_R/2+T_R+T_P)*51 + (T_R/2+T_R+T_P*0,2)$

   t = 5 + (5+10+2)*51 + (5+10+2*0,2) = 5 + 867 + 15,4 = 887,4 ms

10. (4) Neka datoteka veličine 50 MB kompaktno je smještana na disku koji koristi UNIX particiju s veličinom bloka od 4 KB. Prvi blok datoteke nalazi se na disku u bloku rednog broja 2000. Navesti dio opisnika datoteke koji opisuje njen smještaj i skicirati sve ostale strukture podataka koje opisuju smještaj sadržaja te datoteke na disku. Veličina kazaljke na neki blok jest 32 bita.

    50 MB / 4 KB = 50*1024 KB/ 4 KB = 12800 blokova

    prvih 10 kazaljki je u opisniku datoteke s vrijednostima 2000 – 2009

    11. kazaljka pokazuje na dodatni blok na disku u koji može stati: 4 KB / 32 bita = 1024 dodatne kazaljke

    treba nam 12800-10=12790 > 1024 pa se taj blok u potpunosti iskoristi i treba nam i 12. kazaljka

    12. kazaljka je dvostruko indirektna, pokazuje na blok u kojem su kazaljke na blokove s kazaljama

    opisati treba još 12790 – 1024 = 11766 blokova

    11766/1024 = 11 i nešto => 11 punih blokova s kazaljkama i 11766 – 11*1024 = 502 kazaljke u 12. bloku