Drugi kolokvij iz predmeta *Operacijski sustavi* (rješenja)
10. veljače 2026.

1. (2) Kako procesor prihvaća zahtjev za prekid (prije poziva prekidnog potprograma)?

   1. zabrani prekidanje
   2. prebaci se u prekidni način rada
   3. PC i SR spremi na stog
   4. u PC stavi adresu prekidnog potprograma

2. (2) U kojim se sve stanjima može naći dretva (u jednostavnom modelu)?

   aktivna, pripravna, blokirana (na semaforu, na UI napravi, odgođena)

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

   – potpuni zastoj
   – izgladnjivanje

4. (2) Zašto je raspoređivanje podjelom vremena bolje od raspoređivanja po redu prispijeća za normalne poslove?

   – pravedniji način raspoređivanja
   – kraći poslovi prije dođu na red i budu posluženi, ne čekaju kraj dugog posla

5. (2) Usporediti dinamičko upravljanje memorijom i straničenje.

upravljanje	prednosti	nedostatci
dinamičko	zaštita, korištenje pomoćnog spremnika	fragmentacija, ne mogu se pokretati veliki programi
straničenje	zaštita, veliki programi mogu raditi (bez da su cijeli u memoriji), efikasno korištenje memorije i pomoćnog spremnika	složen sklop, usporenje zbog promašaja

6. (2) Navesti osnovne jedinice podataka s 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 (ili samo redni broj sektora uz LBA)
datotečni sustav	blok (ili isto točno: datoteka)	redni broj bloka na particiji (tj. ime datoteke, put do datoteke kroz direktorije)

7. (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 paralelnim korištenjem diskova)
   zalihost – sustav radi i ako se neki disk pokvari

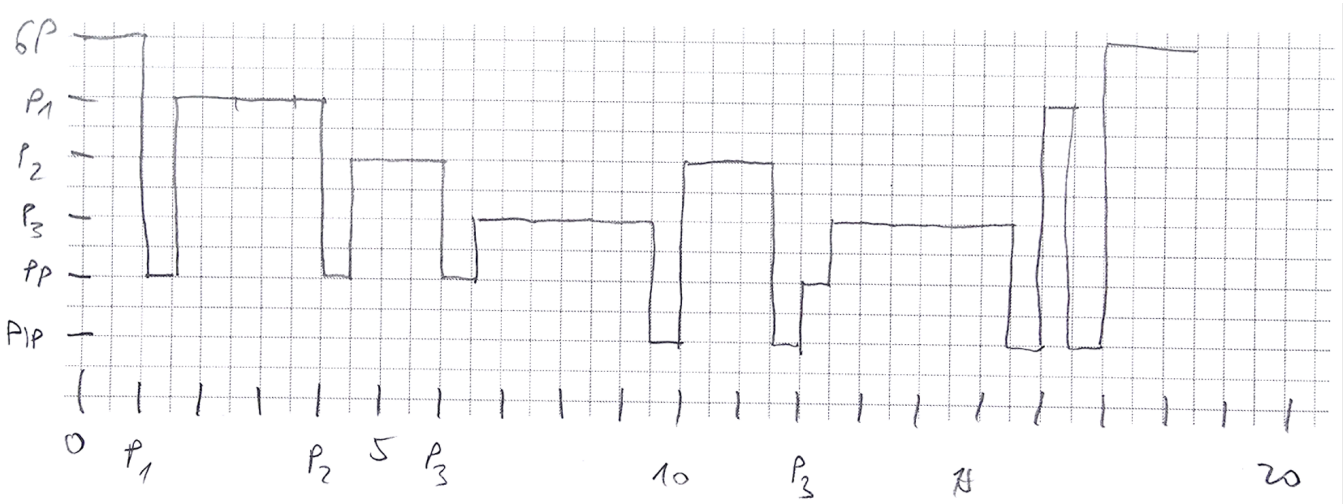
8. Za ostvarenje pohrane u nekom sustavu na raspolaganju stoji 20 diskova, svaki kapaciteta 5 TB. Sustavu je potrebno (dovoljno) 50 TB diskovna prostora.
   a) (1) Ako su potrebne maksimalne performanse i dovoljna je zaštita od kvara jednog diska, koji RAID treba odabrati (0, 1, 5 ili 6) i koliko diskova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

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

   b) (1) Ako je potrebna maksimalna pouzdanost koji RAID treba odabrati (0, 1, 5 ili 6) i koliko diskova koristiti?

      RAID 6, minimalan broj diskova (manja vjerojatnost kvara): (N-2)*5 >= 50 => N=12 diskova

9. (3) U nekom sustavu sa sklopom za prihvat prekida javljaju se prekidi: P_1 u 1., P_2 u 4. te P_3 u 6. ms i i 12. Prioritet prekida je određen 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 3 ms. Grafički prikazati aktivnosti procesora u glavnom programu (GP), procedurama za obradu prekida (P_i) te procedurama za prihvat prekida (PP) i povratak iz prekida (PiP).



10. U nekom sustavu nalaze se dretve A, B, C, D i E s prioritetima p(A)=1, p(B)=2, p(C)=3, p(D)=1 i p(E)=2. 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 $t = 5$ 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

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

    b) (2) dva procesora

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

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

str.	**indeks okvira**	**zast. prisutnosti**
0	0xABCD	1
1	0xDCBA	1
2	-	0
3	-	0
4	0xEF00	1

Tablica 2. Pretvorba adresa

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

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. Na NTFS particiji je pohranjena datoteka: 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 je 16 kB.

a) (2) Napisati tablicu s opisom smještaja (barem dio koji je zadan) te navesti ukupni broj redaka te tablice.

VCN	**LCN**	#
1	10001	640
641	100001	320
961	?	1
962	?	1
itd.	?	1

a)
10MB: 10 MB / 16 KB = 10*1024/16 = 10*64 = 640 blokova

5 MB: 320 blokova

100 MB: 6400 blokova

b) (2) U kojem bloku na disku se nalazi 777. blok datoteke?

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

13. (4) U nekom sustavu koji koristi straničenje javljaju se zahtjevi za stranicama redom: 1 8 2 3 8 2 4 8 3 7 1 2 8 3 2 8 5 3. Ako za te zahjeve stoje na raspolaganju četiri okvira pokazati rad algoritam zamjene stranica LRU – Least-Recently-Used (izbaciti stranicu koja se najdulje nije koristila).

1	8	2	3	8	2	4	8	3	7	1	2	8	3	2	8	5	3
(1)	1	1	1	|	|	(4)	|	|	4	(1)	1	1	1	|	|	(5)	
–	(8)	8	8	[8]	|	8	[8]	|	8	8	(2)	2	2	(2)	|	2	|
–	–	(2)	2	|	[2]	2	|	|	(7)	7	7	7	(3)	|	|	3	[3]
–	–	–	(3)	|	|	3	|	[3]	3	3	3	(8)	8	|	[8]	8	|

Ukupan broj promašaja: 11/18

14. (6) U nekom poslužiteljskom centru zahtjevi korisnika obrađuju se na slijedeći način:
    - zahtjev zaprima glavna dretva (`glavna()`) sa `z=primi()`
    - glavna dretva odbacuje zahtjev ako ih je već 100 u sustavu
    - glavna dretva stavlja zahtjev u red `R1` ako je klijent bitan (`z.prio=VIP`) ili je u sustavu manje od 10 zahtjeva; inače ga stavlja u R2
    - dretve tipa `radna1()` obrađuju zahtjeve iz `R1`, a dretve tipa `radna2()` iz reda `R2`.

    Napisati kôd dretvi uz korištenje semafora ili monitora za sinkronizaciju (proširiti kôd ispod).

```
dretva glavna() {
   ponavljaj {
      z = primi();
      ...
         stavi_u_R1(z)
      ...
         stavi_u_R2(z)
   }
}
```

```
dretva radna1() {
   ponavljaj {
      ...
      z = uzmi_R1();
      ...
      obradi(z)
      ...
   }
}
```

```
dretva radna2() {
   ponavljaj {
      ...
      z = uzmi_R2();
      ...
      obradi(z)
      ...
   }
}
```

Rješenje s monitorima:

```
dretva glavna() {
  ponavljaj {
    z = primi()
    mutex_lock(m)
    ako je broj < 100 {
      broj++
      ako je z.prio==VIP ILI broj<10{
        stavi_u_R1(z)
        r1++
        cond_signal(red1)
      inače {
        stavi_u_R2(z)
        r2++
        cond_signal(red2)
      }
    }
    mutex_unlock(m)
  }
}
```

```
dretva radna1() {
  ponavljaj {
    mutex_lock(m)
    dok je r1 == 0
      cond_wait(red1, m)
    z = uzmi_R1()
    r1--
    mutex_unlock(m)
    obradi(z)
    mutex_lock(m)
    broj--
    mutex_unlock(m)
  }
}

m - monitor
red1, red2 - redovi
broj = r1 = r2 = 0
```

```
dretva radna2() {
  ponavljaj {
    mutex_lock(m)
    dok je r2 == 0
      cond_wait(red2, m)
    z = uzmi_R2()
    r2--
    mutex_unlock(m)
    obradi(z)
    mutex_lock(m)
    broj--
    mutex_unlock(m)
  }
}
```

Rješenje sa semaforima:

```
dretva glavna() {
  ponavljaj {
    z = primi()
    ČekajBSEM(KO)
    ako je broj < 100 {
      broj++
      ako je z.prio==VIP ILI broj<10{
        stavi_u_R1(z)
        r1++
        PostaviOSEM(Red1)
      inače {
        stavi_u_R2(z)
        r2++
        PostaviOSEM(Red2)
      }
    }
    PostaviBSEM(KO)
  }
}
```

```
dretva radna1() {
  ponavljaj {
    ČekajOSEM(Red1)
    ČekajBSEM(KO)
    z = uzmi_R1()
    r1--
    PostaviBSEM(KO)

    obradi(z)

    ČekajBSEM(KO)
    broj--
    PostaviBSEM(KO)
  }
}
```

```
dretva radna2() {
  ponavljaj {
    ČekajOSEM(Red2)
    ČekajBSEM(KO)
    z = uzmi_R2()
    r1--
    PostaviBSEM(KO)

    obradi(z)

    ČekajBSEM(KO)
    broj--
    PostaviBSEM(KO)
  }
}
```

KO - binarni semafor s poč.vr. 1
Red1 i Red2 opći, s poč.vr. 0