Ispit iz predmeta *Operacijski sustavi*

Rješenja

25. lipnja 2020.

1. (4) Od navedenih funkcija, zaokružiti ona koja spadaju u sučelje operacijskog sustava.

   atoi memset mpz_nextprime [printf] [pthread_create]

   [pthread_setschedparam] rand [sem_wait] sqrt

2. (4) Napisati kod u C-u koji će postaviti bit 10 varijable `B` u nulu (obrisati taj bit) samo ako su bitovi 7 i 15 varijable `A` postavljeni (jedinice). Na raspolaganju su uobičajeni C operatori za rad s bitovima: `&`, `|`, `<<`, `>>`, `^` (xor), `~` (okreće sve bitove).

```
if ( (A & (1<<7)) && (A & (1<<15)) //ili ((A>>7)&1)&&((A>>15)&1)
    B = B & ~(1<<10);
//ili za jedan broj manje, umjesto 7 - 6, itd.
```

3. (4) Nadopuniti kod tako da predstavlja Lamportov algoritam međusobnog isključivanja.

```
uđi_u_KO (I) {
  ULAZ[I] = 1
  BROJ[I] = max(BROJ[1], ..., BROJ[N]) + 1
  ULAZ[I] = 0
  za J = 1 do N {
    dok je ( ULAZ[J] == 1    )
      ;
    dok je ( BROJ[J] != 0
      && (BROJ[J] < BROJ[I] || (BROJ[J] == BROJ[I] && J < I)))
      ;
  }
}
izađi_iz_KO (I) { BROJ[I] = 0  }
```

4. (4) Početna vrijednost semafora `s` je 0. Nakon nekog vremena pet dretvi X pozove `sem_post(&s)` te drugih sedam dretvi Y pozove `sem_wait(&s)`.

   a) Kolika je vrijednost semafora nakon tih poziva? Odgovor: ___

   b) Jesu li neke dretve blokirane tim pozivima? Ako jesu, koje i koliko njih?

```
a) 0
b) dvije dretve Y su blokirane
```

5. (4) Postavljanje načina raspoređivanja SCHED_RR za dretvu zahtjeva da je program pokrenuo povlašteni korisnik (root). Obrazložiti zašto je to tako napravljeno.

```
U "normalnim" sustavima, gotovo sve dretve, kako one pokrenute od
korisničkih programa tako i one pokrenute od operacijskog sustava
(razni servisi) su "normalne dretve" (npr. na UNIX-u se raspored
uju prema SCHED_OTHER, na Windowsima su u klasi NORMAL_PRIORITY_
CLASS).
Dretva koja se raspoređuje prema SCHED_RR (ili REALTIME_PRIORITY_
CLASS na Windowsima) spada u kategoriju vremenski kritičnih dretvi
 i kao takva ima veći prioritet od svih "normalnih" dretvi. Greška
 u takvoj dretvi (npr. beskonačna petlja) će "blokirati" cijeli
sustav. Stoga se traži da takve dretve mogu pokretati samo
administratori, a ne obični korisnici.
```

6. (4) Na vrhu stoga nalazi se vrijednost 10. U programskom brojilu je vrijednost 40000. U memoriji na adresi 40000 nalazi se instrukcija `CALL 50000` koja zauzima 4 okteta. Što će biti na vrhu stoga, a što u programskom brojilu nakon izvođenja te instrukcije?

   na vrhu stoga 40004
   u PC 50000

7. (4) Sklop za prihvat prekida (ne pristupni sklop UI naprave!) ima minimalno dva registra. Koji su to registri i čemu služe?

   KZ – kopije zastavica – koji zahtjevi za prekid čekaju
   TP – tekući prioritet – što je u obradi

8. (4) Kojim se mehanizmom pozivaju jezgrine funkije? Zašto baš njime?

   - prekidi
   - prekidima se iz korisničkog načina rada prelazi u privilegirani prihvatom prekida zabranjeni su daljnji prekidi, pa je jezgrina funkcija kritični odsječak na jednoprocesorskim sustavima (na vi šeprocesorskim treba dodati i radno čekanje s OGRADA_JEZGRE)

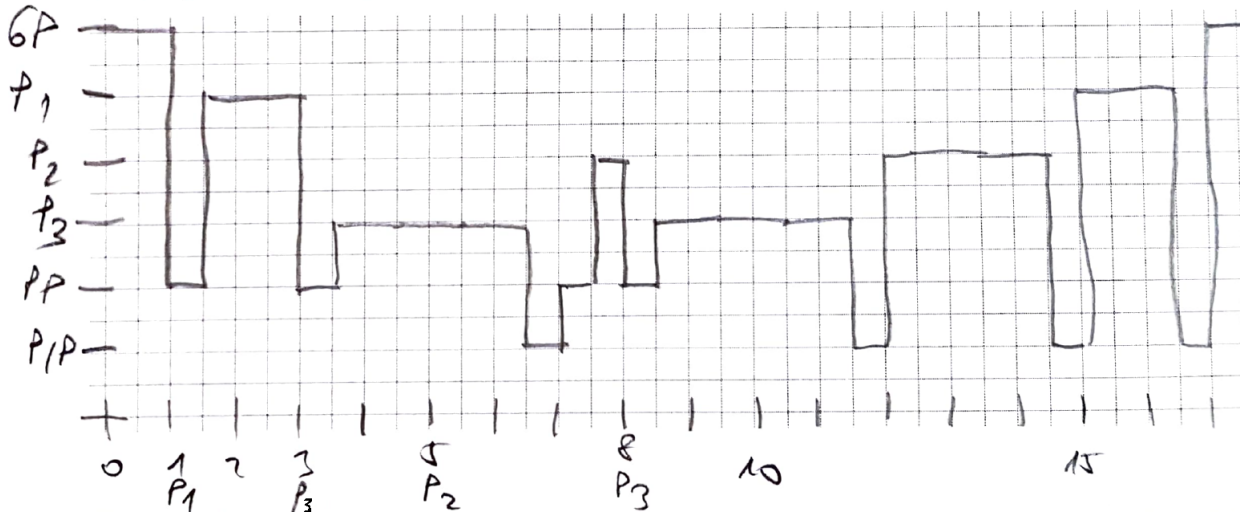
9. (4) U nekom sustavu za rad u stvarnom vremenu dvije dretve razmjenjuju podatke veličine 36 B. Koji mehanizam razmjene podataka je najbolje koristiti u ovom slučaju? Obrazložiti.

   - poruke
   - samo jedna jedna jezgrina finkcija je dovoljna za jednu razmjenu

10. (4) Dva najčešća oblika virtualizacije su virtualizacija na razini operacijskog sustava (npr. VMware, VirtualBox) te virtualizacija na razini sklopovlje (npr. VMware ESX/ESXi, Xen Project, Oracle VM Server, Microsoft Hyper-V). Zašto je ovaj drugi mehanizam učinkovitiji?

    virtualna računala kod virtualizacije na razini sklopovlja zaobilaze OS domaćina što nije slučaj s prvim načinom virtualizacije, gdje domaćin dodatno trpši vrijeme za svaku UI operaciju kao i na privilegirane operacije

11. (7) U nekom sustavu sa sklopom za prihvat prekida javljaju se prekidi: P_1 u 1. ms, P_2 u 5. te P_3 u 3. i 8. 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 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).



12. (7) U nekom sustavu operacijski sustav koristi dinamičko upravljanje spremnikom. Za procese je na raspolaganju 200 MB. Procesi koji se pokreću su već pripremljeni na pomoćnom spremniku. Događaji u sustavu su slijedeći: zahtjev za pokretanje P_1 (75 MB), zahtjev za pokretanje P_2 (25 MB), zahtjev za pokretanje P_3 (50 MB), P_1 završava, zahtjev za pokretanje P_4 (100 MB), zahtjev za pokretanje P_5 50 MB, P_2 završava, P_5 završava, P_3 završava, P_4 završava. Ukoliko se neki zahtjev za pokretanje ne može poslužiti u tom trenutku, on čeka dok se oslobodi dovoljno memorije za njega. Pokazati stanje memorije nakon svakog od navedenih događaja. Sustav ne koristi defragmentaciju memorije.

```
crtica = 25 MB
t = 0    --------
+P1      111-----
+P2      1112----
+P3      111233--
-P1      ---233--
+P4      ---233-- (ne stane, memorija je fragmentirana!)
+P5      ---23355 (P5 u manju rupu !!!)
-P2      ----3355
         44443355
-P5      444433--
-P3      4444----
-P4      --------
```

13. (7) Program na poslužitelju treba se ostvariti s dvije dretve *zaprimi* te osam dretvi *obradi*. Cikličke dretve *zaprimi* zaprimaju zahtjeve od klijenata sa $z = Zaprimi()$ te ga stavljaju u red neobrađenih zahtjeva sa *StaviURed(z)*. Za obradu pojedinog zahtjeva treba obaviti nekoliko operacija. Broj tih operacija koje treba napraviti zapisan je u zahtjevu u elementu *z.N*. Dretva koja obavlja zadnju operaciju nad zahtjevom miče taj zahtjev iz reda s *MakniPrvi()*. Po potrebi proširiti zahtjev dodatnim poljima (npr. *z.n*, *z.m*). Dohvat prvog zahtjeva iz reda obavlja se pozivom $x = DohvatiPrvi()$ – funkcija ne miče zahtjev iz reda već samo daje referencu na prvi objekt u redu. Cikličke dretve *obradi* obavljaju po jednu operaciju nad prvim zahtjevom iz reda sa *Obradi(x)* u svakoj svojoj iteraciji, ili čekaju na novi zahtjev kada je red prazan. Po završetku svih operacija nad nekim zahtjevom klijentu treba poslati odgovor sa *Pošalji(x)*. Slanje treba napraviti dretva koja zadnja završi s obradom operacije nad zahtjevom (što ne mora biti dretva koja je makla zahtjev iz reda!). Rad s redom zahtjeva treba zaštiti od paralelnog pristupa, kao i ažuriranje zahtjeva nakon dovršetka svake operacije. Sinkronizirati dretve semaforima.

```
dretva zaprimi {
    ponavljaj {
        z = Zaprimi()
        z.n = z.m = z.N

        ČekajBSEM(KO)
        StaviURed(z)
        za i = 1 do z.N radi
            PostaviOSEM(OP)
        PostaviBSEM(KO)
    }
}

Poč. vrijednosti:
BSEM[KO].v = 1
OSEM[OP].v = 0
```

```
dretva obradi {
    ponavljaj {
        ČekajOSEM(OP)
        ČekajBSEM(KO)
        x = DohvatiPrvi()
        ako je x.n == 1 tada
            MakniPrvi()
        x.n--
        PostaviBSEM(KO)
        Obradi(x)
        ČekajBSEM(KO)
        x.m--
        ako je x.m == 0 tada
            Pošalji(x)
        PostaviBSEM(KO)
    }
}
```

14. (7) Zadan je sljedeći algoritam:

```
za i = 1 do N-1 korak 2 radi //1 3 5 ...
    za j = 1 do N radi
        x = A[i,j]
        y = A[i+1,j]
        A[i+1,j] = max(x,y)
        A[i,j] = min(x,y)
```

a) Odrediti redoslijed zahtjeva za stranicama veličine N riječi (N je paran).

Pokazati rad strategije LRU ako sustav za zadanu matricu ima na raspolaganju

b) 1 okvir

c) 2 okvira.

Zanemariti promašaje radi dohvata instrukcija te korištenja pomoćnih varijabli.

```
Traži se redoslijed zahtjeva za stranicama, ne (samo) za
elementima matrice!

- jedan redak matrice stane u jednu stranicu
- redak "i" stane u stranicu "i"

Zahtjevi:
       elementi matrice
      (indeksi)            stranice
i=1
j=1     1-1 2-1 2-1 1-1      1 2 2 1
j=2     1-2 2-2 2-2 1-2      1 2 2 1
...
j=N     1-N 2-N 2-N 1-N      1 2 2 1
---------
i=3
j=1     3-1 4-1 4-1 3-1      3 4 4 3
j=2     3-2 4-2 4-2 3-2      3 4 4 3
...
j=N     3-N 4-N 4-N 3-N      3 4 4 3
---------
....
---------
i=N-1
i=3
j=1     (N-1)-1 N-1 N-1 (N-1)-1      (N-1) N N (N-1)
j=2     (N-1)-2 N-2 N-2 (N-1)-2      (N-1) N N (N-1)
...
j=N     (N-1)-N N-N N-N (N-1)-N      (N-1) N N (N-1)

a) Niz zahtjeva:
1 2 2 1 1 2 2 1 ... 1 2 2 1 | 3 4 4 3 ... | ... | N-1 N N N-1 ...

b) 1 okvir
1 2 2 1 1 2 2 1 ... 1 2 2 1 | 3 4 4 3 ... | ... | N-1 N N N-1 ...
1 2 | 1 | 2 | 1 ... | 2 | 1 | 3 4 | 3 ...
svaki neparni je pogodak osim prvog
za svaki "j" imamo 4/2 = 2 promašaja
```

```
za svaki "i" imamo N takvih + 1 (prvi) => 2N+1
imamo N/2 takvij "i" te je broj promašaja: N/2 * (2N+1)

c) 2 okvira
1 2 2 1 1 2 2 1 ... 1 2 2 1 | 3 4 4 3 ... | ... | N-1 N N N-1 ...
1 1 x X dalje sve pogotci   | 3 3 x ...
- 2 X x                     | 2 4 X ...

Za svaki "i" po dva promašaja: N/2 * 2 = N promašaja
```

15. (7) Tri istovjetna poslužitelja zaprimaju i obrađuju poslove različitih svojstava. Prvi prosječno zaprima 100 z/s s prosječnim trajanjem zadržavanja posla u sustavu od 5 ms. Drugi prosječno zaprima 20 z/s s prosječnim trajanjem zadržavanja posla u sustavu od 12,5 ms. Treći prosječno zaprima 10 z/s s prosječnim trajanjem zadržavanja posla u sustavu od 20 ms. U slučaju nestanka struje treći poslužitelj preuzima sve poslove (prva dva se gase), ali radi duljeg rada na baterijama smanjuje si frekvenciju procesora na 77% uobičajene. Koliko će biti prosječno vrijeme zadržavanja poslova u sustavu kada nestane struje, kada sve poslove obavlja treći poslužitelj? Pretpostaviti Poissonovu razdiobu za dolaske poslova te eksponencijalnu za trajanje obrade.

```
Koraci u ovom rješenju:
1. izračunati opterećenja svakog poslužitelja prije nestanka
   struje
2. izračunati koliki bi bio beta na trećem poslužitelju kad bi sve
   radio sam
3. ažurirati beta za smanjenje frekvencije
4. izračunati T

1.
a1 = 100 z/s  -- b1 = a1 + 1 / T1 = 100 z/s + 1/5ms = 300 z/s
T1 = 5 ms     -- r1 = a1 / b1 = 100/300 = 1/3

a2 = 20 z/s   -- b2 = 20 z/s + 1/12,5ms = 100 z/s
T2 = 12,5 ms  -- r2 = 20/100 = 1/5

a3 = 10 z/s   -- b3 = 10 z/s + 1/20ms = 60 z/s
T3 = 20 ms    -- r3 = 10/60 = 1/6

2.
a3' = a1 + a2 + a3 = 130 z/s (svi poslovi dolaze na 3. posl.)
obzirom da su poslužitelji isti, opterećenje samo zbrojimo
r3' = r1 + r2 + r3 = 1/3+1/5+1/6 = 21/30 = 7/10 = 0,7 (70%)

b3' = a3' / r3' = (100 + 20 + 10) / (7/10) = 1300/7 z/s

3.
b3" = b3' * 0,77 = 1300/7 * 0,77 = 1300 * 0,11 = 13*11 = 143

4.
T3" = 1 / (b3" - a3") = 1 / (143 - 130) = 1/13 s (=76,9 ms)
```

16. (7) Neka datotke pohranjena je na disku na slijedeći način: prvih 1000 blokova datoteke nalazi se u blokovima diska 75001-76000, idućih 500 blokova datoteke u blokovima diska 300001-300500 te zadnjih 5000 blokova datoteke u blokovima diska 753001-758000.

a) Opisati tu datoteku (njen smještaje) prema NTFS datotečnom sustavu (VCN/LCN/#).

b) Ako se na disku pokvari blok 757575, koji blok datoteke će se izgubiti?

a)

VCN	LCN	#
1	75001	1000
1001	300001	500
1501	753001	50000

b) blok diska 757575 se pokvari – koji je to blok datoteke?
iz tablice se vidi da je to u zadnjem dijelu datoteke
koji koristi blokove diska: 753001-758000

obzirom da je 753001. blok diska 1501. blok datoteke:
757575-753001 = 4574
1501 + 4574 = 6075 => izgubljen je 6075. blok datoteke