Elektrotehnički fakultet u Beogradu
Katedra za računarsku tehniku i informatiku

Predmet: Operativni sistemi 2
Nastavnik: prof. dr Dragan Milićev
Odsek: Softversko inženjerstvo, Računarska tehnika i informatika
Kolokvijum: Drugi, februar 2026.
Datum: 13. 3. 2026.

Drugi kolokvijum iz Operativnih sistema 2

Kandidat: _____

Broj indeksa: _____ *E-mail:*_____

Kolokvijum traje 90 minuta. Dozvoljeno je korišćenje literature.

Zadatak 1 _____/10 *Zadatak 3* _____/10
Zadatak 2 _____/10

Ukupno: _____/30 = _____%

Napomena: Ukoliko u zadatku nešto nije dovoljno precizno definisano, student treba da uvede razumnu pretpostavku, da je uokviri (da bi se lakše prepoznala prilikom ocenjivanja) i da nastavi da izgrađuje preostali deo svog odgovora na temeljima uvedene pretpostavke. Ocenjivanje unutar potpitanja je po sistemu "sve ili ništa", odnosno nema parcijalnih poena. Kod pitanja koja imaju ponuđene odgovore treba **samo zaokružiti** jedan odgovor. Na ostala pitanja odgovarati **čitko, kratko i precizno**.

1. (10 poena) Mrtva blokada

Neki softverski sistem organizovan je u *n* modula. Svaki modul ima svoj interfejs i ostali moduli mogu da pristupaju samo tom interfejsu. Interfejs svakog modula sastoji se od potprograma, od kojih neki predstavljaju kritične sekcije zaštićene semaforima. Potprogrami istog modula koje pozivaju ovi interfejsni potprogrami nemaju zaključavanje niti u sebi imaju ugnežđene kritične sekcije zaštićene ključevima, ali potprogrami iz jednog modula mogu pozivati potprograme iz interfejsa drugih modula od kojih dati modul zavisi.

a)(5) Ako se ove zavisnosti između modula predstave grafom u kom su moduli predstavljeni čvorovima a zavisnosti usmerenim granama, dokazati da je uslov da je ovaj graf DAG (*directed acyclic graph*, usmereni graf bez petlji) dovoljan, ali ne i potreban uslov da ovaj sistem ne može ući u mrtvu blokadu zaključavanjem na opisanim kritičnim sekcijama.

b)(5) Definisati potreban i dovoljan uslov da ovaj sistem ne može ući u mrtvu blokadu.

Rešenje:

2. (10 poena) Virtuelna memorija

Neki sistem ima PMT u dva nivoa. PMT prvog nivoa predstavljen je tipom `PMT0_t`, a PMT drugog nivoa tipom `PMT1_t` datim dole. Ulaz u PMT prvog nivoa sadrži pokazivač na PMT drugog nivoa ili *null* ako ona nije alocirana. Operativni sistem sve PMT drugog nivoa svih procesa alocira složene u niz `pmt1` veličine `PMT1_count` elemenata (pregradaka). Uporedo sa tom strukturom, operativni sistem vodi matricu `refBits` iste strukture - `PMT1_count` puta `PMT1_size` ulaza tipa `unsigned int` koji čuvaju dodatne bite referenciranja odgovarajućih stranica (`refBits[i][j]` odgovara deskriptoru `pmt1[i][j]`). Ulazi u ovom nizu za stranice koje nisu nikada korišćene imaju vrednost 0.

Napisati funkciju `getVictimPage` koja vraća broj stranice - žrtve izabrane za izbacivanje procesa na čiju PMT prvog nivoa ukazuje parametar; žrtva se bira lokalno samo za taj proces.

```
const size_t PMT0_size = ..., PMT1_count = ..., PMT1_size = ...;
typedef PMT1_t* PMT0_t[PMT0_size];
typedef PMTEntry PMT1_t[PMT1_size];
PMT1_t pmt1[PMT1_count];
unsigned int refBits[PMT1_count][PMT1_size];

size_t getVictimPage (PMT0_t* pmt0);
```

Rešenje:

3. (10 poena) Alokacija memorije

Napisati funkciju koja određuje veličinu ploče u broju stranica tako da fragmentacija unutar ploče bude minimalna. Deklaracija funkcije je:
`int calculateSlabSize(int pageSize, int objectSize, int metaDataSize, int maxSlabSize).`
Parametri funkcije su redom veličina stranice, veličine objekta za ploču, veličina režijskih podataka za ploču na početku ploče i maksimalna veličina ploče. Prve tri veličine su zadate u bajtovima, dok je poslednja u broju stranica.

Rešenje: