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, decembar 2023.
Datum: 03. 12. 2023.

Prvi 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 = _____%

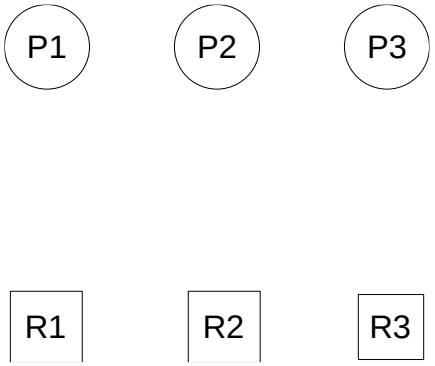
Napomena: Ukoliko u zadatku nešto nije dovoljno precizno definisano, student treba da uvede razumnu pretpostavku, da je uokviri (da bi se lakše prepoznala prilikom ocenjivanja) i da nastavi da izgrađuje preostali deo svog odgovora na temeljima uvedene pretpostavke. Ocenjivanje unutar potpitanja je po 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

U nekom sistemu koji izbegava mrtvu blokadu algoritmom zasnovanim na grafu alokacije postoje tri aktivna procesa, P1, P2 i P3, koji su najavili korišćenje resursa na sledeći način: P1 je najavio korišćenje resursa R1 i R2, P2 je najavio korišćenje resursa R1, R2 i R3, a P3 je najavio korišćenje resursa R2 i R3. Procesi redom izvršavaju sledeće operacije (+ označava zahtev za resursom, - označava oslobađanje resursa):
P2+R1, P3+R3, P1+R2, P2+R3 (1), P3-R3 (2), P2-R1 (3)

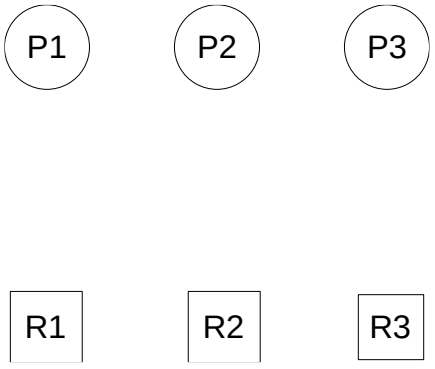
Za mesta u sekvenci označena brojevima od (1) do (3) nacrtati graf zauzeća resursa u označenom stanju sistema (nakon što je sistem u potpunosti obradio prethodnu operaciju, uključujući i zadovoljenje eventualnih zahteva procesa koji su čekali na resurse) i navesti procese koji su suspendovani u tom stanju:

1)



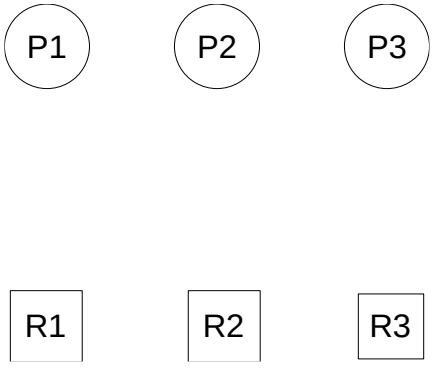
Suspendovani su procesi: _____

2)



Suspendovani su procesi: _____

3)



Suspendovani su procesi: _____

2. (10 poena) Virtuelna memorija

Neki sistem ima PMT u dva nivoa. Operativni sistem sve PMT drugog nivoa svih procesa alocira složene u niz `pmt1` veličine `PMT1_COUNT` elemenata (pregradaka). Svaki PMT drugog nivoa ima `PMT1_SIZE` ulaza tipa `PMTEntry`. U svakom tom ulazu (deskriptoru stranice) najniži bit je bit referenciranja, a susedni bit zaprljanosti. Ukoliko u jedan pregradak niza `pmt1` nije alociran PMT drugog nivoa, u prvom deskriptoru tog PMT nalazi se vrednost `FREE_PMT_SLOT`. Ukoliko stranica nije alocirana, njen deskriptor ima vrednost `FREE_PMT_ENTRY`. Uporedo sa tom strukturom, operativni sistem vodi matricu `refBits` iste strukture - `PMT1_COUNT` puta `PMT1_SIZE` 32-bitnih ulaza koji čuvaju dodatne bite referenciranja odgovarajućih stranica (`refBits[i][j]` odgovara deskriptoru `pmt1[i][j]`).

Napisati funkcije `updateRefBits` i `getVictim` koje operativni sistem koristi pri periodičnom ažuriranju dodatnih bita referenciranja, odnosno pri izboru stranice žrtve za zamenu, respektivno.

```
typedef unsigned PMTEntry;
const int PMT1_COUNT = ..., PMT1_SIZE = ...;
extern PMTEntry pmt1[PMT1_COUNT][PMT1_SIZE];
const PMTEntry FREE_PMT_SLOT = ~0, FREE_PMT_ENTRY = 0;
extern uint32_t refBits[PMT1_COUNT][PMT1_SIZE];

void updateRefBits ();
PMTEntry* getVictim ();
```

Rešenje:

3. (10 poena)

Sistem opisan u zadatku 2 poseduje zaštitu od pojave zaglavljivanja (engl. *thrashing*), tako što povremeno izvršava operaciju `freeUnusedPages` koja iz memorije izbacuje sve one stranice koje nisu korišćene u najskorija 24 intervala za koje se pamte biti referenciranja i koje nisu zaprljane (ne moraju da se snimaju prilikom izbacivanja). Ova funkcija vraća broj takvih izbačenih stranica. Jedna stranica, definisana ulazom `pmt1[pmt][pmtEntry]` oslobađa se funkcijom `discardPage` koja je na raspolaganju. Implementirati funkciju `freeUnusedPages`.

```
int discardPage(int pmt, int pmtEntry);
int freeUnusedPages ();
```

Rešenje: