Elektrotehnički fakultet u Beogradu
Katedra za računarsku tehniku i informatiku

Predmet: Operativni sistemi 2 (SI3OS2, IR3OS2)
Nastavnik: prof. dr Dragan Milićev
Odsek: Softversko inženjerstvo, Računarska tehnika i informatika
Kolokvijum: Drugi, septembar 2012.
Datum: 21.8.2012.

Drugi kolokvijum iz Operativnih sistema 2

Kandidat: _____

Broj indeksa: _____ *E-mail:*_____

Kolokvijum traje 1,5 sat. 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 svaki proces i svaki resurs ima svoj jedinstveni identifikator (tipa `unsigned int`), a zauzeće resursa od strane procesa prati se u matrici `resourceAlloc` u kojoj vrste označavaju procese, a kolone resurse. U toj matrici vrednost 1 u ćeliji (*p*, *r*) označava da je proces sa identifikatorom *p* zauzeo resurs sa identifikatorom *r*, a vrednost 0 označava da proces *p* nije zauzeo resurs *r*. Mrtva blokada sprečava se tako što se procesu dozvoljava zauzimanje resursa samo u opadajućem redosledu vrednosti identifikatora resursa. Zato se, kada proces *p* zatraži resurs *r*, poziva operacija `allocate(p,r)`:

```
const unsigned MAXPROC = ...; // Maximum number of processes
const unsigned MAXRES = ...;  // Maximum number of resources
extern unsigned numOfProc;    // Actual number of processes
extern unsigned numOfRes;     // Actual number of resources

int resourceAlloc[MAXPROC][MAXRES];

int allocate (unsigned pid, unsigned rid);
```

Ova operacija proverava da li se procesu *p* može i sme dodeliti resurs *r* i zauzima ga, ako može. U tom slučaju ova operacija treba da vrati 1. Ako se procesu ne može dodeliti resurs, treba vratiti 0. Implementirati operaciju `allocate`.

Rešenje:

2. (10 poena) Upravljanje memorijom

Za izbor stranice za zamenu u nekom sistemu koristi se aproksimacija LRU algoritma sa dodatnim bitima referenciranja. Svakoj stranici pridružen je 8-bitni registar sa sačuvanim ranijim bitima referenciranja koji se pomeraju udesno. Posmatra se grupa od četiri stranice označene sa 1-4 i sledeća sekvenca njihovog referenciranja (simbol ↓ označava periodični prekid na koji se ažuriraju registri referenciranja):

1, 2, 3, 4, 2, 4, 1, ↓, 2, 1, 3, 1, ↓, 1, 3, 4, 1, ↓, 2, 1, 3, 2, 1, ↓, 4, 2, 4, 2, 4, ↓, 1, 3, 2, 1, 2, 3, ↓

Dati heksadecimalne vrednosti registara za ove četiri stranice nakon ove sekvence.
Ako se nakon ove sekvence mora zameniti neka od ovih stranica, koja će to stranica biti?

Rešenje:

3. (10 poena) Upravljanje memorijom

U jezgru nekog operativnog sistema primenjuje tehnika izbegavanja pojave *thrashing* praćenjem aproksimacije radnog skupa. Sistem periodično prebrojava bite referenciranja korišćenih stranica svakog procesa i tim brojem aproksimira veličinu radnog skupa tog procesa. U PCB svakog procesa polje `pmt` ukazuje na PMT procesa. PMT je niz od `NumOfPages` deskriptora stranica. Svaki deskriptor je tipa `unsigned int`, a u njemu je najviši bit bit referenciranja.

Implementirati funkciju

```
unsigned long workingSetSize (PCB* pcb);
```

koja se poziva periodično i koja treba da prebroji postavljene bite referenciranja datog procesa i vrati taj broj.

Rešenje: