Elektrotehnički fakultet u Beogradu
Katedra za računarsku tehniku i informatiku

Predmet: Operativni sistemi 1 (SI2OS1, IR2OS1)
Nastavnik: prof. dr Dragan Milićev
Odsek: Softversko inženjerstvo, Računarska tehnika i informatika
Kolokvijum: Drugi, maj 2012.
Datum: 5.5.2012.

Drugi kolokvijum iz Operativnih sistema 1

Kandidat: _____

Broj indeksa: _____ *E-mail:*_____

Kolokvijum traje 1,5 sat. Dozvoljeno je korišćenje literature.

Zadatak 1	_____/10	*Zadatak 3*	_____/10
Zadatak 2	_____/10	*Zadatak 4*	_____/10

Ukupno: _____/40 = _____% = _____/15

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)

Dva uporedna kooperativna procesa A i B razmenjuju podatke tako što proces A izračunava vrednost deljene promenljive `a` pozivom svoje privatne procedure `compute_a`. Tu vrednost promenljive `a` koristi proces B tako što na osnovu nje izračunava vrednost deljene promenljive `b` pozivom svoje privatne procedure `compute_b`. Tu vrednost promenljive `b` potom koristi proces A za izračunavanje nove vrednosti promenljive `a` i tako ciklično. Implementirati ove procese korišćenjem uposlenog čekanja za uslovnu sinhronizaciju. Inicijalno, proces A može odmah da izračuna početnu vrednost za `a` na osnovu statički inicijalizovane vrednosti za `b`, dok proces B ne može da izračuna prvu vrednost `b` dok ne dobije prvu izračunatu vrednost `a`.

Rešenje:

2. (10 poena)

Data je bibliotečna funkcija namenjena podršci optimističkom pristupu međusobnom isključenju bez eksplicitnog zaključavanja:

```
int cmpxchg(void** ptr, void* oldValue, void* newValue);
```

Ova funkcija kao prvi argument (`ptr`) prima adresu lokacije u kojoj se nalazi neki pokazivač bilo kog tipa (`void*`). Ona atomično poredi vrednost pokazivača koji je dostavljen kao drugi argument (`oldValue`) sa vrednošću na lokaciji na koju ukazuje `ptr`, i ako su te vrednosti iste, u lokaciju na koju ukazuje `ptr` upisuje vrednost datu trećim argumentom (`newValue`) i vraća 1; u suprotnom, ako su ove vrednosti različite, ne radi ništa, već samo vraća 0. Atomičnost je obezbeđena implementacijom pomoću odgovarajuće mašinske instrukcije.

Struktura `Record` predstavlja zapis (jedan element) jednostruko ulančane liste. U toj strukturi polje `next` ukazuje na sledeći element u listi.

Korišćenjem date operacije `cmpxchg()` implementirati funkciju:

```
void insert (Record** head, Record* e);
```

Ova funkcija prima argument koji predstavlja adresu pokazivača na prvi element liste (adresu lokacije u kojoj je glava liste), a kao drugi argument dobija pokazivač na novi element u listi koga treba da umetne na početak date liste. Lista je deljena između više procesa, pa ova funkcija treba da bude sigurna za uporedne pozive iz više procesa, s tim da međusobno isključenje treba obezbediti optimističkom strategijom bez eksplicitnog zaključavanja. Zapis na koga ukazuje drugi element (`e`) je privatan samo za pozivajući proces (drugi procesi mu ne pristupaju pre umetanja u listu).

Rešenje:

3. (10 poena)

Neki program treba da izračuna proizvod dve ogromne matrice A[$m \times k$]×B[$k \times n$]=C[$m \times n$], gde su dimenzije matrica veoma velike (i po nekoliko miliona). Predložiti i precizno opisati kako biste primenili tehniku preklopa (*overlays*) u ovom programu. Posebno objasniti način smeštanja matrica u preklope i redosled učitavanja u preklope.

Odgovor:

4. (10 poena)

Virtuelni adresni prostor nekog sistema je 16EB (eksabajta, $1E=2^{60}$) i organizovan je stranično, adresibilna jedinica je bajt, a stranica je veličine 4KB. Fizički adresni prostor je veličine 4TB (terabajta). PMT (*page map table*) je organizovana u dva nivoa, s tim da su i broj ulaza, kao i širina ulaza u PMT prvog i drugog nivoa isti (PMT oba nivoa su iste veličine). PMT oba nivoa smeštaju se u memoriju uvek poravnate na fizički okvir, odnosno uvek počinju na početku okvira. Zbog toga se u ulazu prvog nivoa čuva samo broj okvira u kom počinje PMT drugog nivoa, dok se preostali biti do celog broja bajtova u ulazu ne koriste; vrednost 0 u svim bitima označava da preslikavanje nije dozvoljeno. U jednom ulazu PMT drugog nivoa čuva se broj okvira u koji se stranica preslikava i još 2 bita koja koduju prava pristupa (00 – nedozvoljen pristup, stranica nije u memoriji, 01 – dozvoljeno samo izvršavanje instrukcije, 10 – dozvoljeno samo čitanje podataka, 11 – dozvoljeno i čitanje i upis podataka); jedan ulaz u PMT drugog nivoa zauzima minimalan, ali ceo broj bajtova.

Kada sistem kreira nov proces, ne učitava inicijalno ni jednu njegovu stranicu, niti alocira ijednu PMT drugog nivoa, već samo alocira PMT prvog nivoa, čije sve ulaze inicijalizuje nulama. Stranice se potom dohvataju na zahtev, tokom izvršavanja procesa, kada se po potrebi alociraju i PMT drugog nivoa.

Odgovoriti na sledeća pitanja uz detaljna obrazloženja postupka.

a)(3) Prikazati logičku strukturu virtuelne i fizičke adrese i označiti veličinu svakog polja.

Odgovor:

b)(3) Koliko memorije minimalno zauzima PMT alocirana za proces prilikom njegovog kreiranja?

Odgovor:

c)(4) Koliko ukupno memorije zauzimaju sve PMT alocirane za proces koji je u dosadašnjem toku izvršavanja koristio najnižih 512GB svog virtuelnog adresnog prostora?

Odgovor:

Izrada: