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, april 2014.
Datum: 27.4.2014.

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

Ukupno: _____/30 = _____% = _____/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)

Modifikovati operaciju `wait` klase `Semaphore` u školskom jezgru tako da ima izmenjenu deklaraciju datu dole sa sledećim dodatnim mogućnostima i izmenjenim ponašanjem:

- ukoliko je vrednost argumenta `toBlock` različita od 0, operacija se ponaša na standardan način i vraća 1 ako se pozivajuća nit blokirala (suspendovala), a 0 ako nije;
- ukoliko je vrednost argumenta `toBlock` jednaka 0, ako je vrednost semafora nula (ili manja od 0), pozivajuća nit se neće blokirati, vrednost semafora se neće promeniti, a operacija će odmah vratiti -1; inače, ukoliko je vrednost semafora veća od nule, operacija se ponaša na standardan način (i vraća 0 pošto se nit nije blokirala).

```
int Semaphore::wait (int toBlock);
```

Rešenje:

2. (10 poena)

Neki sistem primenjuje kontinualnu alokaciju memorije. Slobodni fragmenti dvostruko su ulančani u listu na čiji prvi element ukazuje `fmem_head`. Svaki slobodni fragment predstavljen je strukturom `FreeMem` koja je smeštena na sam početak tog slobodnog fragmenta:

```
struct FreeMem {
  FreeMem* next; // Next in the list
  FreeMem* prev; // Previous in the list
  size_t size;   // Size of the free fragment
};
```

Implementirati funkciju

```
int mem_alloc(void* address, size_t by);
```

Ova funkcija pokušava da pronađe slobodan fragment na adresi datoj prvim argumentom i da iz njega alocira deo veličine date drugim argumentom. Ukoliko na datoj adresi ne počinje slobodan fragment ili on nije dovoljne veličine, ova funkcija vraća -1. Ukoliko na datoj adresi postoji slobodan fragment dovoljne veličine, ova funkcija ažurira listu slobodnih framenata na sledeći način:

- ukoliko bi iza alociranog dela preostao fragment koji je manji ili jednak veličini strukture `FreeMem`, ceo taj slobodni fragment se alocira (ne ostavlja se ostatak koji je premali za evidenciju) i izbacuje iz liste slobodnih;
- u suprotnom, preostali deo se ostavlja kao slobodan fragment.

U oba slučaja, funkcija vraća veličinu stvarno alociranog dela (veće ili jednako traženoj veličini).

Rešenje:

3. (10 poena)

Virtuelni adresni prostor nekog sistema je 1EB (eksabajt, $1E=2^{60}$) i organizovan je straničnо, adresibilna jedinica je bajt, a stranica je veličine 4KB. Fizički adresni prostor je veličine 4TB (terabajt). PMT (*page map table*) je organizovana u tri nivoa, s tim da su i broj ulaza, kao i širina ulaza u PMT sva tri nivoa isti (PMT sva tri nivoa su iste veličine). PMT sva tri nivoa smeštaju se u memoriju uvek poravnate na fizički okvir, odnosno uvek počinju na početku okvira. Zbog toga se u jednom ulazu u PMT prvog/drugog nivoa čuva samo broj okvira u kom počinje PMT drugog/trećeg 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 trećeg 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 trećeg nivoa zauzima minimalan, ali ceo broj bajtova.

Kada sistem kreira nov proces, ne učitava inicijalno nijednu njegovu stranicu, niti alocira ijednu PMT drugog i trećeg 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 i trećeg 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 adresirao najnižih 1GB svog virtuelnog adresnog prostora?

Odgovor:

Izrada: