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, septembar 2015.
Datum: 4.9.2015.

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)

U školskom jezgru modifikuje se (uopštava) koncept brojačkog semafora prodržan klasom `Semaphore` na sledeći način. Operacija `wait()` prima jedan nenegativan celobrojni argument *n* sa podrazumevanom vrednošću 1 i sa sledećim značenjem. Trenutna nenegativna vrednost semafora *v* predstavlja broj raspoloživih „žetona“. Operacijom *wait* pozivajuća nit „traži“ *n* „žetona“ (kod standardnog brojačkog semafora *n* je uvek podrazumevano 1). Ako je trenutna vrednost *v* semafora veća ili jednaka argumentu *n* operacije *wait*, ta vrednost *v* će biti umanjena za *n*, a nit će nastaviti izvršavanje bez blokade, jer je „dobila“ svih traženih *n* „žetona“. U suprotnom, ova nit će „uzeti“ *v* „žetona“, i čekaće blokirana na semaforu dok se operacijama *signal* ne pojavi još *n*-*v* žetona; kada se to dogodi, nit može nastaviti sa izvršavanjem (jer je dobila svih traženih *n* „žetona“). Operacija *signal* „obezbeđuje“ uvek jedan „žeton“, kao i kod standardnog brojačkog semafora.

```
class Semaphore {
public:
  Semaphore (unsigned int init=1);
  void wait(unsigned int n=1);
  void signal();
  int val ();
};
```

Za podršku ovoj implementaciji, u klasi `Thread` postoji nenegativno celobrojno polje `waiting` koje pokazuje na koliko još preostalih „žetona“ čeka data nit, ukoliko je blokirana na nekom semaforu. Osim toga, klasa `Queue` kojom se implementira FIFO red čekanja na semaforu ima operaciju `first()` koja vraća prvu nit u tom redu, ako je ima (0 ako je red prazan), bez izbacivanja te niti iz reda. Pomoćne operacije `block()` i `deblock()` klase `Semaphore` su iste kao i u postojećoj implementaciji školskog jezgra.

Dati izmenjenu implementaciju operacija `wait()` i `signal()`.

Rešenje:

2. (10 poena)

Neki sistem primenjuje kontinualnu alokaciju memorije. Slobodni fragmenti dvostruko su ulančani u listu na čiji prvi element ukazuje `fmem_head`. Fragmenti su ulančani u listu po rastućem redosledu svojih početnih adresa. 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_free(char* address, size_t size);
```

Ova funkcija treba da dealocira zauzeti kontinualni segment memorije na datoj adresi i date veličine, uz eventualno (po potrebi) spajanje novooslobođenog fragmenta sa susednim slobodnim fragmentima ispred i iza njega.

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 i drugog nivoa, u najnižim bitima, 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, u najnižim bitima čuva se broj okvira u koji se stranica preslikava i još 2 bita sdesna 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.

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

Odgovor:

b)(7) U PCB procesa polje `pmt1` tipa `unsigned` sadrži broj okvira od koga počinje PMT prvog nivoa. Na raspolaganju je sledeća funkcija koja okvir sa datim brojem proglašava slobodnim (upisuje ga u evidenciju slobodnih okvira):

```
void deallocateFrame (unsigned frameNumber);
```

Implementirati sledeću funkciju koja se koristi prilikom gašenja procesa ili njegovog izbacivanja iz memorije (*swap out*) i koja oslobađa sve okvire koje proces koristi:

```
void deallocateAllFrames (PCB* pcb);
```

Pretpostavlja se da proces ne deli stranice sa drugim procesima. Mašinska reč, kao i tip `int` su veličine 32 bita, dok su pokazivači, kao i tip `long` veličine 64 bita.

Rešenje: