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, novembar 2011.
Datum: 3.12.2011.

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**.

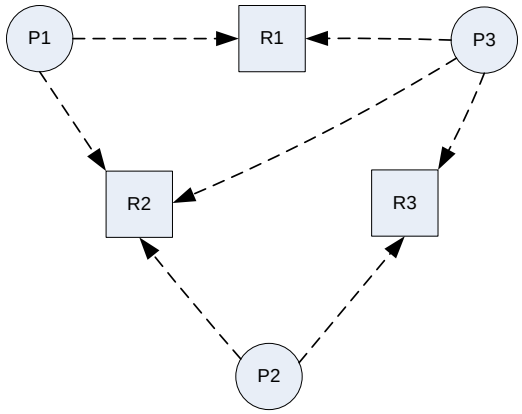
1. (10 poena) Mrtva blokada

Neki sistem primenjuje algoritam izbegavanja mrtve blokade (*deadlock avoidance*) pomoću grafa alokacije. Na slici je dat graf početnog stanja sistema sa tri procesa i tri resursa.

a)(3) Nacrtati graf stanja sistema nakon sledeće sekvence:
P2.request(R3), P1.request(R1), P2.request(R2), P3.request(R3)

b)(3) Nakon sekvence date pod a), P2 oslobađa resurs R3. Nacrtati graf stanja sistema nakon ove operacije.

c)(4) Nakon toga, sekvenca se nastavlja ovako: P1.request(R2), P2.release(R2). Nacrtati graf stanja nakon toga.



2. (10 poena) Upravljanje memorijom

Za izbor stranice za zamenu u nekom sistemu koristi se algoritam „davanja nove šanse“ ili „časovnika“ (*second-chance, clock algorithm*). Primenjuje se lokalna politika zamene stranice unutar istog procesa. U deskriptoru stranice u PMT, koji je tipa `unsigned long int`, najviši bit je bit referenciranja. U posebnom nizu `pagefifo` svaki element odgovara jednoj stranici i sadrži indeks ulaza u tom nizu koji predstavlja sledeću stranicu u kružnoj FIFO strukturi stranica uređenih po redosledu učitavanja.

```
struct PCB {
  ...
  unsigned int clockHand;
  unsigned long* pmt; // Pointer to the PMT
  unsigned int* pagefifo; // Pointer to the FIFO page table
};
```

Implementirati funkciju `getVictimPage(PCB*)` koja vraća broj stranice koju treba zameniti po ovom algoritmu zamene, za proces sa datim PCB.

Rešenje:

3. (10 poena) Upravljanje memorijom

U jezgru nekog operativnog sistema primenjuje se sistem ploča (*slab allocator*) za alokaciju struktura za potrebe jezgra. Za alokaciju nove ploče kada u kešu više nema slobodnih slotova koristi se niži sloj koji implementira *buddy* alokator.
U nastavku su date su delimične definicije klasa `Cache` i `Slab` i implementacije nekih njihovih operacija. Slotovi za alokaciju su tipa `X`. Struktura `Cache` predstavlja keš i čuva pokazivač `headSlab` na prvu ploču u kešu; ploče u kešu su ulančane u jednostruku listu. Strukura `Slab` predstavlja ploču. U njoj su pokazivač na sledeću ploču istog keša `nextSlab`, kao i pokazivač `freeSlot` na prvi slobodan slot u nizu `slots` svih slotova u ploči. Slobodni slotovi su ulančani u jednostruku listu preko pokazivača koji se smeštaju u sam slot, na njegov početak. Inicijalizaciju jedne ploče vrši dati konstruktor.

```
const int numOfSlotsInSlab = ...;

class Cache {
public:
  ...
  X* alloc();
private:
  friend class Slab;
  Slab* headSlab;
}

class Slab {
public:
  Slab (Cache* ownerCache);
  static void* operator new (size_t s) { return buddy_alloc(s); }
  static void  operator delete (void* p) { buddy_free(p,sizeof(Slab)); }
private:
  friend class Cache;
  Slab* nextSlab;
  X* freeSlot;
  X slots[numOfSlotsInSlab];
};

Slab::Slab(Cache* c) {
    this->nextSlab=c->headSlab;
    c->headSlab=this;
    this->freeSlot=&this->slots;
    for (int i=0; i<numOfSlotsInSlab-1; i++)
      *(X**)(&slots[i])=&slots[i+1];
    *(X**)(&slots[numOfSlotsInSlab-1])=0;
}
```

Implementirati operaciju `Cache::alloc()` koja treba da alocira jedan slobodan slot `X`. Nije potrebno optimizovati alokaciju tako da se slot traži najpre u delimično popunjenim pločama, pa tek u praznoj, već se može prosto vratiti prvi slobodan slot na koga se naiđe.

Rešenje: