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 2013.
Datum: 20.8.2013.

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 su tri procesa (P_1, P_2, P_3) i tri različite instance resursa (R_1, R_2, R_3). Primenjuje se izbegavanje mrtve blokade praćenjem zauzeća resursa pomoću grafa, a resurs se dodeljuje procesu čim je to moguće i dozvoljeno. Posmatra se sledeća sekvenca operacija:

P2.request(R3), P1.request(R1), P2.request(R2), P3.request(R3), P2.release(R3), P1.request(R2), P3.request(R1), P2.release(R2), P1.release(R1), P3.release(R3), P1.release(R2), P3.release(R1)

Procesi najavljuju korišćenje onih i samo onih resursa koje zauzimaju u datoj sekvenci.

a)(7) Do kog dela se ova sekvenca može izvršiti baš u datom redosledu, ako se primenjuje izbegavanje mrtve blokade? Nacrtati graf zauzeća resursa u tom trenutku.

b)(3) Nakon koje operacije će proces koji prvi nije dobio resurs odmah kad ga je tražio dobiti taj resurs?

Rešenje:

2. (10 poena) Upravljanje memorijom

Za izbor stranice za zamenu u nekom sistemu koristi se algoritam časovnika (engl. *clock algorithm*). Svaka učitana stranica nekog procesa opisana je strukturom `PageDescr` čija je definicija data dole. Polje `page` u ovoj strukturi čuva broj stranice koju opisuje struktura, a polje `ref` čuva vrednost bita referenciranja. Ovo polje `ref` ažurira posebna periodična rutina sistema na osnovu bita referenciranja koje održava hardver i koja nije relevantna ovde. Za svaki proces, ove strukture su ulančane u dvostruko ulančanu kružnu listu, kao podrška algoritmu zamene. Za ovu svrhu služi klasa `PageClock` čija je implementacija data dole; jedan objekat ove klase implementira listu učitanih stranica i algoritam izbacivanja za jedan proces.

Realizovati operaciju `PageClock::removeVictim()` koja treba da odabere stranicu za izbacivanje, izbaci njenu strukturu iz ulančane liste i vrati broj odabrane stranice. Ukoliko je lista stranica prazna, treba vratiti -1 kao kod greške.

```
typedef unsigned int PageNo;

struct PageDescr {
  PageNo page; // Page number
  int ref; // Reference bit
  PageDecr* next;
  PageDescr* prev;

  PageDescr (PageNo pg, PageDescr* nxt, PageDescr* prv) : page(pg),ref(0) {
    this->prev=prv;
    if (this->prev) this->prev->next=this;
    else this->prev=this;
    this->next=nxt;
    if (this->next) this->next->prev=this;
    else this->next=this;
  }

  void remove () {
    if (this->prev) this->prev->next=this->next;
    if (this->next) this->next->prev=this->prev;
  }
};

class PageClock {
public:
  PageClock () : hand(0) {}
  void   addPage (PageNo page);  // Adds the loaded page
  PageNo removeVictim ();  // Returns and removes the victim page
private:
  PageDescr* hand; // Clock hand (cursor)
};

void PageClock::addPage (PageNo pg) {
  if (hand==0)
    hand = new PageDescr(pg,0,0);
  else
    new PageDescr(pg,hand->prev,hand);
}
```

Rešenje:

3. (10 poena) Memorijski preslikani fajlovi

U nekom operativnom sistemu podržan je koncept memorijski preslikanih fajlova (engl. *memory mapped files*).
Podsistem za upravljanje fajlovima u svom interfejsu, između ostalog, poseduje i funkciju:

```
int fread(FHandle fh, void* buffer, unsigned offset, unsigned size);
```

koja iz sadržaja fajla sa datom ručkom, počev od date pozicije `offset` (u bajtovima, od početnog bajta 0 sadržaja fajla), učitava niz bajtova date dužine `size` u zadati bafer `buffer`.

Svaki logički segment memorije nekog procesa koji se preslikava u fajl opisan je strukturom `MMFSegment` čija je delimična definicija data dole. Polja `pgLow` i `pgHigh` predstavljaju brojeve prve i poslednje stranice u tom segmentu. Polje `fh` je ručka fajla u koji je preslikan dati segment memorije. Ova polja postavlja inicijalizaciona procedura koja se poziva u sistemskom pozivu kojim se traži preslikavanje dela memorije u fajl.

Realizovati funkciju `pgLoad()` koju poziva deo sistema za zamenu stranica kada želi da učita traženu stranicu `pg`, koja pripada datom memorijskom segmentu `mmf`, u već alocirani okvir fizičke memorije `frame`.

Sve pomenute funkcije vraćaju celobrojni status: negativnu vrednost u slučaju greške, 0 u slučaju ispravnog završetka. Pretpostaviti da je sadržaj fajla dovoljno veliki da se u njega preslika ceo dati segment (to obezbeđuje inicijalizacija preslikavanja).

```
typedef unsigned int PageNo;
typedef ... FHandle;
const unsigned int PAGESIZE = ...;  // Page size (in bytes)

struct MMFSegment {
  PageNo pgLow, pgHigh;
  FHandle fh;
  ...
}

int pgLoad(MMFSegment* mmf, PageNo pg, void* frame) ;
```

Rešenje: