Elektrotehnički fakultet u Beogradu
Katedra za računarsku tehniku i informatiku

Predmet: Operativni sistemi 2
Nastavnik: prof. dr Dragan Milićev
Odsek: Softversko inženjerstvo, Računarska tehnika i informatika
Kolokvijum: Integralni, januar 2021.
Datum: 21. 1. 2021.

Integralni kolokvijum iz Operativnih sistema 2

Kandidat: _____

Broj indeksa: _____ *E-mail:*_____

Kolokvijum traje 2 sata. Dozvoljeno je korišćenje literature.

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

Ukupno: _____/40 = _____%

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 nekom sistemu koristi se raspoređivanje procesa sa prioritetom u opsegu `0..MaxPri` (niža vrednost označava viši prioritet). U skupu procesa istog prioriteta raspoređivanje je *round-robin*. U strukturi `PCB` postoje sledeća polja:

- Polje `curPri` tipa `unsigned` predstavlja tekući, dinamički prioritet procesa, a polje `defPri` podrazumevani, početni prioritet procesa koji se ne menja tokom izvršavanja.
- Polje `next` je pokazivač na sledeći PCB u listi (za ulančavanje u jednostruke liste).

Kada se proces vraća u red spremnih, dinamički prioritet mu se menja za 1, ali samo u opsegu `defPri ± PriMargin` (tekući prioritet „pliva“ do rastojanja konstantne margine `PriMargin` oko podrazumevanog prioriteta), u zavisnosti od toga da li proces u red spremnih dolazi iz stanja suspenzije ili zato što mu je istekao vremenski odsečak (sami odredite na koji način).

Klasa `Scheduler`, čiji je interfejs dat dole, realizuje opisani raspoređivač spremnih procesa. Implementirati u potpunosti ovu klasu tako da i operacija dodavanja novog spremnog procesa `put()` i operacija uzimanja spremnog procesa koji je na redu za izvršavanje `get()` budu ograničene po vremenu izvršavanja vremenom koje ne zavisi od broja spremnih procesa (kompleksnost $O(1)$). U slučaju da nema drugih spremnih procesa, operacija `get()` vraća *null*. Drugi argument operacije `put()` govori da li je proces postao spreman zato što je prethodno bio blokiran (`wasBlocked=true`) ili mu je istekao vremenski kvantum (`wasBlocked=false`).

```
class Scheduler {
public:
  Scheduler ();
  PCB* get ();
  void put (PCB*, bool wasBlocked);
};
```

Rešenje:

2. (10 poena)

Neki sistem detektuje mrtvu blokadu modifikacijom bankarevog algoritma. U sistemu je aktivno četiri procesa (*P*1..*P*4). U posmatranom stanju procesi su zauzeli resurse kao što je dato u matrici *Alllocation*, a postavili su zahteve za resurse i čekaju da te resurse dobiju kao što je dato u matrici *Request*; procesi koji ne zahtevaju nijedan resurs su izvršivi. Da li je ovaj sistem u mrtvoj blokadi? Pokazati detaljan postupak dolaska do odgovora. Ako jeste, koji procesi učestvuju u mrtvoj blokadi? Ako nije, navesti barem jedan redosled kojim se procesi mogu izvršavati do kraja.

Allocation

	A	*B*	*C*
*P*1	1	0	1
*P*2	0	1	1
*P*3	2	1	1
*P*4	1	0	0

Request

	A	*B*	*C*
*P*1	0	1	3
*P*2	3	0	1
*P*3	0	0	0
*P*4	2	1	0

Available

A	*B*	*C*
0	1	1

Rešenje:

3. (10 poena)

Neki sistem ima alokator paranjaka (*buddy*) koji realizuje klasa `Buddy` čija je delimična implementacija data dole. Alociraju se segmenti veličine 2^i susednih blokova veličine `BLOCK_SIZE` u jedinicama `sizeof(char)`, gde je $0 \leq i <$ `BUCKET_SIZE`. Svaki *i*-ti element niza `bucket` sadrži listu slobodnih segmenata veličine 2^i blokova.

a)(5) Implementirati pomoćnu operaciju:

```
void Buddy::split (char* seg, int upper, int lower);
```

koja slobodan segment veličine `2^upper` blokova na koji ukazuje parametar `seg`, a koji nije ni u jednoj listi niza `bucket`, deli na dva dela, višu polovinu smešta na odgovarajuće mesto u niz `bucket`, a niži deo deli dalje na dva jednaka dela itd. sve do dela veličine `2^lower` blokova. Parametar `upper` je veći ili jednak parametru `lower`; ako je jednak, funkcija ne treba da radi ništa.

b)(5) Implementirati operaciju

```
void* Buddy::alloc (int size);
```

koja alocira segment veličine `2^size` blokova; ako nema slobodnog mesta, treba da vrati 0.

```
class Buddy {
public:
  Buddy (void* p) { bucket[BUCKET_SIZE-1].put((char*)p); }
  void* alloc (int size);
  ...
private:
  void split (char* seg, int upper, int lower);

  class List {
  public:
    List () : head(0) {}
    void put (char* p) { *(char**)p = head;  head = p; }
    char* get () { char* p = head; if (head) head = *(char**)head; return p; }
  private:
    char* head;
  };

  List bucket[BUCKET_SIZE];
};
```

Rešenje:

4. (10 poena)

Napisati dva programa na programskom jeziku C koji međusobno komuniciraju koristeći isključivo imenovane cevi operativnog sistema Linux. Greške koje vraćaju funkcije za rad sa imenovanim cevima nije potrebno obrađivati.

Nad prvim programom pokreće se samo jedan proces iz komandne linije. Smatrati da će proces biti pokrenut značajno pre drugih procesa. Taj proces treba da prima poruke od procesa koji su pokrenuti nad drugim programom. Poruka je string veličine 50 bajtova. Primljenu poruku prvi program treba da pretvori u ceo broj koristeći datu funkciju `makeHash` sa deklaracijom:

```
int makeHash(char *msg);
```

Ceo broj treba da vrati procesu koji mu je poslao poruku kao rezultat. Program treba da prima poruke sve dok ne primi poruku `"STOP"`.

Nad drugim programom se pokreće proizvoljno mnogo procesa iz komandne linije. Prilikom pokretanja kao parametar mu se prosleđuje pozitivan ceo broj. Svaki pokrenuti proces će dobiti drugačiji broj od svih drugih procesa. Onaj proces koji primi broj 100 treba da pošalje poruku `"STOP"`. Proces treba da kreira poruku korišćenjem date funkcije `getMsg` sa deklaracijom:

```
void getMsg(char *msg);
```

Rezultat koji mu vrati prvi program treba da ispiše na standardni izlaz i da se završi.

Rešenje: