Elektrotehnički fakultet u Beogradu
Katedra za računarsku tehniku i informatiku

Predmet: Operativni sistemi 1
Nastavnik: prof. dr Dragan Milićev
Odsek: Softversko inženjerstvo, Računarska tehnika i informatika
Kolokvijum: Drugi, jun 2018.
Datum: 20. 6. 2018.

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 klasu `Semapore` u školskom jezgru dodata je statička funkcija članica

```
int Semaphore::waitOr(Semaphore*,Semaphore*);
```

koja čeka na dva semafora po *ili* uslovu, tj. dok bilo koji od dva semafora ne postane veći od 0. Ukoliko je taj uslov ispunjen, ova operacija smanjuje samo jedan od njih za 1 (nikada oba, čak i ako su oba veća od 0), i vraća 1 ili 2, ukazujući na to koji od njih je umanjila za 1 (prvi ili drugi).

Potrebno je implementirati ograničeni bafer čiji je interfejs dat dole. Ovom baferu dostavljaju se uporedo, od strane dve vrste proizvođača, elementi tipa `int` operacijom `put1`, odnosno elementi tipa `double` operacijom `put2`. Ove elemente treba smeštati u dva različita interna bafera (za svaki tip elementa po jedan). Sa druge strane, potrošač operacijom `get` treba da dobije bilo koji od dve vrste elemenata, koji god je na raspolaganju. Ova operacija `get` treba da postavi uzetu vrednost u jedan od dva objekta na koji ukazuju argumenti, i da vrati vrednost 1 ili 2, u zavisnosti koji od ta dva objekta je postavio vrednost.

```
class BoundedBuffer {
public:
  BoundedBuffer ();

  void  put1 (int);
  void  put2 (double);
  int   get  (int*, double*);
};
```

Rešenje:

2. (10 poena)

Neki sistem primenjuje kontinualnu alokaciju memorije sa *best-fit* algoritmom. Zapisi o slobodnim delovima memorije organizovani su u ulančanu listu uređenu neopadajuće po veličini slobodnih delova memorije. U svakom zapisu je informacija o adresi početka slobodnog dela memorije i njegovoj veličini. Dat je sadržaj ove liste u nekom trenutku (sve vrednosti su heksadecimalne).

Zapis broj	Adresa početka	Veličina
1	3510	20
2	3680	30
3	3470	50
4	35A0	90

a)(4) Prikazati tu listu nakon alokacije dela memorije veličine 40h.

Rešenje:

b)(3) Prikazati tu listu nakon što se, posle akcije pod a), izvrši oslobađanje dela memorije veličine 70h, na adresi 3530h.

Rešenje:

c)(3) Prikazati tu listu nakon što se, posle akcije pod b), izvrši kompakcija slobodnog prostora pomeranjem alociranih delova na sam početak prvog slobodnog dela. Iza poslednjeg slobodnog dela nalazi se memorijski prostor koji se ne koristi u ovoj alokaciji.

Rešenje:

3. (10 poena)

Virtuelni adresni prostor nekog sistema je 4GB i organizovan je stranično, adresibilna jedinica je bajt, a stranica je veličine 16KB. Fizički adresni prostor je veličine 16MB. PMT (*page map table*) je organizovana u dva nivoa, s tim da je broj ulaza u PMT prvog i drugog nivoa isti.

Za svaki proces operativni sistem kreira memorijski kontekst kao skup deskriptora logičkih memorijskih segmenata. Deskriptor segmenta predstavljen je strukturom `SegDesc` u kojoj, između ostalog, postoje sledeća polja:

- `unsigned long startingPage`: početna stranica segmenta u virtuelnom prostoru;
- `unsigned long size`: veličina segmenta izražena u broju stranica;
- `SegDesc *next, *prev`: sledeći i prethodni deskriptor segmenta u dvostruko ulančanoj listi segmenata istog procesa; na prvi segment u listi ukazuje polje `segDesc` u PCB procesa;
- `short rwx`: u tri najniža bita definisana su prava pristupa stranicama datog segmenta, odnosno dozvoljene operacije za taj logički segment.

Kada stranica nije u memoriji ili ne pripada alociranom segmentu, u odgovarajućem ulazu u PMT nalazi se vrednost 0 koja hardveru indikuje da preslikavanje nije moguće; procesor tada generiše izuzetak tipa stranične greške (*page fault*). U slučaju povrede prava pristupa stranici, odnosno nedozvoljene operacije, procesor generiše izuzetak koji naziva *operation denied*. Sistem primenjuje *copy on write* tehniku.

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

Odgovor:

b)(7) Implementirati funkciju `resolveVAddrExc`:

```
enum AddrExcKind {pageFault, opDenied};

int resolveVAddrExc (PCB* pcb, unsigned long vaddr, AddrExcKind kind,
                     short rwx, SegDesc** ret);
```

Ovu funkciju poziva kod kernela kada obrađuje neki izuzetak prilikom adresiranja virtuelne adrese za proces na čiji PCB ukazuje prvi argument, prilikom pristupa adresi datoj drugim argumentom. Tip izuzetka daje argument `kind`, a operaciju koja je pokušana argument `rwx`.

Ukoliko data virtuelna adresa nije regularna (ne pripada definisanom logičkom segmentu), ili tražena operacija nije dozvoljena za tu adresu, ova funkcija treba da vrati rezultat `MEM_ACCESS_FAULT`, na osnovu koga će kernel ugasiti proces. U suprotnom, radi se o slučaju kada je adresiranje dozvoljeno, ali stranica nije u memoriji i treba je učitati, i kada funkcija treba da vrati `LOAD_PAGE`, ili o slučaju kada treba iskopirati stranicu prilikom upisa, i kada funkcija treba da vrati `COPY_ON_WRITE`. Kad god je virtuelna adresa unutar definisanog logičkog segmenta, u izlazni argument na koga ukazuje argument `ret` treba upisati pokazivač na deskriptor segmenta kom pripada data (regularna) virtuelna adresa.

Rešenje: