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, maj 2016.
Datum: 8.5.2016.

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)

Školsko jezgro proširuje se konceptom *mutex* koji predstavlja binarni semafor, poput događaja (*event*), sa istom semantikom operacija *wait* i *signal* kao kod događaja, ali sa sledećim dodatnim ograničenjima koja su u skladu sa namenom upotrebe samo za međusobno isključenje kritične sekcije:

- inicijalna vrednost je uvek 1;
- operaciju *signal* može da pozove samo nit koja je zatvorila ulaz u kritičnu sekciju, odnosno koja je pozvala operaciju *wait*; u suprotnom, operacija *signal* vraća grešku;
- nit koja je već zatvorila *mutex* operacijom *wait*, ne može ponovo izvršiti *wait* na njemu.

Operacije *wait* i *signal* vraćaju celobrojnu vrednost, 0 u slučaju uspeha, negativnu vrednost u slučaju greške. Prikazati implementaciju klase `Mutex`, po uzoru na klasu `Semaphore` prikazanu na predavanjima (ne treba implementirati red čekanja niti, pretpostaviti da ta klasa postoji).

Rešenje:

2. (10 poena)

Neki sistem koristi preklope (*overlays*). Prevodilac i linker u generisanom kodu prevedenog programa koji koristi preklope statički alociraju i adekvatno inicijalizuju sledeće strukture podataka:

- za svaki modul (preklop, *overlay*) postoji sledeći deskriptor; moduli koji se preklapaju imaju istu početnu adresu:

```
struct OverlayDescr {
  const char* filename; // Naziv fajla u kome je binarni sadržaj preklopa
  void* addr; // Adresa u adresnom prostoru procesa na kojoj se nalazi
  bool isLoaded; // Da li je preklop trenutno učitan?
}
```

- tabela svih preklopa-modula:

```
const int numOfOverlays = ...; // Ukupan broj preklopa
OverlayDescr overlays[numOfOverlays];  // Tabela svih preklopa
```

- svakom potprogramu koji se nalazi u nekom preklopu prevodilac pridružuje jedan jedinstveni ceo broj (identifikator), koji predstavlja ulaz u tabelu tih potprograma; svaki ulaz u ovoj tabeli sadrži pokazivač na deskriptor preklopa u kome se nalazi taj potprogram:

```
const int numOfProcs = ...; // Ukupan broj potprograma
OverlayDescr* procedureMap[numOfProcs];  // Tabela potprograma
```

Za svaki poziv potprograma koji se nalazi u nekom preklopu, na uvek istoj i prevodiocu poznatoj adresi u adresnom prostoru procesa, npr. `proc(a,b,c)`, prevodilac generiše kod koji je ekvivalent sledećeg koda:

```
ensureOverlay (proc_id); // proc_id je identifikator za proc
proc(a,b,c); // standardan poziv na poznatoj adresi
```

Funkcija `ensureOverlay(int procID)` treba da obezbedi da je potprogram sa datim identifikatorom prisutan u memoriji, odnosno po potrebi učita njegov preklop. Potrebno je implementirati ovu funkciju, pri čemu je na raspolaganju sistemski poziv koji učitava binarni sadržaj iz fajla sa zadatim imenom na zadatu adresu u adresnom prostoru procesa:

```
void sys_loadBinary(const char* filename, void* address);
```

Rešenje:

3. (10 poena)

U nekom sistemu koristi se straničenje sa PMT u jednom nivou i tehnika *copy-on-write*. Tabela deskriptora alociranih memorijskih segmenata jednog procesa ima sledeći sadržaj:

Segment#	*Starting page*	*Size in pages*	*RWX*
1	00h	2h	001
2	A0h	4h	100
3	C1h	3h	110

PMT ovog procesa u posmatranom stanju ima sledeći sadržaj (svi ulazi koji nisu navedeni imaju sadržaj 000 u polju *RWX*, što hardveru znači da preslikavanje nije dozvoljeno iz bilo kog razloga i da treba generisati *page fault*):

Page#	*Frame#*	*RWX*
00h	210h	
01h	211h	
A0h	212h	
A1h	213h	
A2h	214h	
A3h	215h	
C1h	216h	
C2h	217h	
C3h	218h	

a)(2) U prethodnu tabelu upisati vrednosti u kolonu *RWX*, ako su sve stranice učitane.

b)(4) U opisanom stanju ovaj proces kreira proces-dete pozivom `fork()`. Prikazati PMT procesa-roditelja nakon uspešnog završetka ovog poziva.

c)(4) Nakon toga, proces-roditelj izvršava instrukciju koja upisuje vrednost u lokaciju na stranici C2h. Prikazati PMT oba procesa nakon što je operativni sistem obradio izuzetak koji je nastao tokom izvršavanja ove instrukcije. Pretpostaviti da je prvi slobodan okvir 219h.

Rešenje:

b) PMT procesa-roditelja:

Page#	*Frame#*	*RWX*
00h		
01h		
A0h		
A1h		
A2h		
A3h		
C1h		
C2h		
C3h		

c) PMT procesa-roditelja:

Page#	*Frame#*	*RWX*
00h		
01h		
A0h		
A1h		
A2h		
A3h		
C1h		
C2h		
C3h		

c) PMT procesa-deteta:

Page#	*Frame#*	*RWX*
00h		
01h		
A0h		
A1h		
A2h		
A3h		
C1h		
C2h		
C3h		