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: Prvi, jun 2017.
Datum: 11.6.2017.

Prvi 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)

Date su deklaracije pokazivača preko kojih se može pristupiti registrima jednog kontrolera mrežne kartice, kao i jednog DMA kontrolera koji je direktno spregnut sa tom mrežnom karticom i može prenositi paket veličine `PKT_SIZE` primljen sa mreže u memoriju:

```
typedef volatile unsigned int REG;
REG* ioNetCtrl =...;    // Network device control register
REG* ioNetStatus =...;  // Network device status register
REG* ioNetData =...;    // Network device data register
REG* dmaCtrl =...;      // DMA control register
REG* dmaStatus =...;    // DMA status register
REG* dmaAddr =...;      // DMA buffer address register
REG* dmaCount =...;     // DMA buffer size register
const int BUF_SIZE = ..., PKT_SIZE = ...;
char buffer[BUF_SIZE][PKT_SIZE];
int bufHead, bufTail;
```

Kada stigne jedan paket sa mreže, mrežna kartica generiše prekid procesoru. Tada treba pokrenuti transfer tog paketa sa mrežne kartice u memoriju, korišćenjem DMA kontrolera. DMA kontroler se pokreće upisom konstante `DMA_START` u njegov upravljači registar. Paket veličine `PKT_SIZE` treba smestiti na mesto na koje ukazuje indeks `bufTail` u kružnom baferu `buffer`. Ako je bafer pun, što se vidi tako da je `bufTail==bufHead`, mrežnoj kartici treba javiti da primljeni paket treba da odbaci (tj. da prijemnoj strani treba da javi da prijem nije uspeo), što se postiže upisom konstante `PKT_REJECT` u upravljački registar mrežne kartice.

Napisati kod prekidnih rutina za prekide sa mrežne kartice i DMA kontrolera.

Rešenje:

2. (10 poena)

U školskom jezgru promena konteksta implementirana je korišćenjem date funkcije `yield()`, u sistemskom pozivu `dispatch()` i na svim ostalim mestima na sličan način kao što je dato.

Potrebno je implementirati sistemski poziv (statičku operaciju):

```
void Thread::wait(Thread* forChild=0);
```

kojim pozivajuća nit čeka (suspenduje se ako je potrebno) dok se ne završi nit-dete na koje ukazuje argument, odnosno sve niti-deca koje je ova pozivajuča nit do tada kreirala, ako je ovaj argument *null*. Za te potrebe treba implementirati i sledeće nestatičke funkcije-članice:

- `void Thread::created(Thread* parent)`: poziva je jezgro interno za datu novokreiranu nit (`this`), kada je ta nit kreirana, sa argumentom `parent` koji ukazuje na roditeljsku nit u čijem kontekstu je ova nova nit-dete kreirana;
- `void Thread::completed()`: poziva je jezgro za datu nit (`this`), kada se ta nit završila.

Ukoliko proširujete klasu `Thread` novim članovima, precizno navedite kako.

```
void yield (jmp_buf old, jmp_buf new) {
  if (setjmp(old)==0) longjmp(new,1);
}

void Thread::dispatch () {
  lock();
  jmp_buf old = Thread::running->context;
  Scheduler::put(Thread::running);
  Thread::running = Scheduler::get();
  jmp_buf new = Thread::running->context;
  yield(old,new);
  unlock();
}
```

Rešenje:

3. (10 poena)

Data je klasa `Thread` koja implementira niti kao u školskom jezgru, ali samo sa jednom mogućnošću konstrukcije objekata te klase, kako je dato u njenom interfejsu dole (samo objektno orijentisani API za kreiranje niti).

a)(7) Realizovati klasu `ThreadFnCaller` koja omogućava kreiranje niti nad zadatom globalnom funkcijom koja prima jedan argument tipa `void*` i ne vraća rezultat, tako da ta nit izvršava zadatu funkciju sa zadatim argumentom.

b)(3) Napisati fragment koda koji kreira `N` niti nad funkcijom `fn`, pri čemu *i*-ta nit izvršava tu funkciju sa argumentom koji je dat u *i*-tom elementu niza `args`.

```
class Thread {
public:
  Thread ();
  void start ();
  virtual void run () {}
}:
```

Rešenje: