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, septembar 2014.
Datum: 29.8.2014.

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 nekom operativnom sistemu postoje samo standardni brojački semafori podržani klasom `Semaphore` čiji je interfejs isti kao u školskom jezgru:

```
class Semaphore {
public:
  Semaphore (int init=1);
  void wait();
  void signal();
  int val ();
};
```

Pomoću ovih semafora implementirati koncept događaja (binarnog semafora), podržanog klasom `Event` sa sledećim interfejsom:

```
class Event {
public:
  Event ();
  void wait();
  void signal();
  int val ();
};
```

Rešenje:

2. (10 poena)

Kod za prvi prolaz nekog linkera dat je u nastavku. (U prvom prolazu linker samo prikuplja izvezene simbole, a ne proverava i da li su svi uvezeni simboli i definisani; tu proveru radi prilikom obrade uvezenih simbola u drugom prolazu.) Popuniti izostavljene delove koda označene komentarima `/***1***/` do `/***5***/`.

```
void Linker::firstPass () {
  this.status = OK;
  this.binarySize = /***1***/; //BinarySize of processed output
  for (FileReader::reset(); !FileReader::isDone(); FileReader::next()) {
    ObjectFile* objFile = FileReader::currentObjectFile();
    if (objFile==0) {
      Output::error("Fatal internal error: null pointer exception.");
      exit(-1);
    }
    for (objFile.reset(); !objFile.isDone(); objFile.nextSymbol()) {
      Symbol* sym = objFile->getCurrentSymbol();
      if (sym==0) {
        /***2***/
      }
      if (sym->getKind() == Symbol::export) {
        int offset = sym->getOffset(); // offset in objFile
        offset += /***3***/;
        int status =
          SymbolTable::addSymDef(sym->getName(),offset,objFile->getName());
        if (status==-1) {
          /***4***/
          this.status = ERROR;
        }
      }
    }
    int size = objFile->getBinarySize();
    /***5***/
  }
}
```

Rešenje:

```
/***1***/



/***2***/



/***3***/



/***4***/



/***5***/
```

3. (10 poena)

Virtuelni adresni prostor nekog sistema je 16EB (eksabajt, $1E=2^{60}$) i organizovan je straničnо, adresibilna jedinica je bajt, a stranica je veličine 16KB. Fizički adresni prostor je veličine 1TB (terabajt). PMT (*page map table*) je organizovana u dva nivoa, s tim da su i broj ulaza, kao i širina ulaza u PMT prvog i drugog nivoa isti (PMT oba nivoa su iste veličine). PMT oba nivoa smeštaju se u memoriju uvek poravnate na fizički okvir, odnosno uvek počinju na početku okvira. Zbog toga se u ulazu prvog nivoa čuva samo broj okvira u kom počinje PMT drugog nivoa, dok se preostali biti do celog broja bajtova u ulazu ne koriste; vrednost 0 u svim bitima označava da preslikavanje nije dozvoljeno. U jednom ulazu PMT drugog nivoa čuva se broj okvira u koji se stranica preslikava i još 2 najniža bita koja koduju prava pristupa (00 – nedozvoljen pristup, stranica nije u memoriji, 01 – dozvoljeno samo izvršavanje instrukcije, 10 – dozvoljeno samo čitanje podataka, 11 – dozvoljeno i čitanje i upis podataka), dok se ostali biti ne koriste. Jedan ulaz u PMT prvog i drugog nivoa zauzima minimalan, ali ceo broj bajtova. Kada sistem kreira nov proces, ne učitava inicijalno ni jednu njegovu stranicu, niti alocira ijednu PMT drugog nivoa, već samo alocira PMT prvog nivoa, čije sve ulaze inicijalizuje nulama. Stranice se potom dohvataju na zahtev, tokom izvršavanja procesa, kada se po potrebi alociraju i PMT drugog nivoa.

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

Odgovor:

b)(7) Implementirati sledeću funkciju:

```
void setPMTEntry (unsigned* pmt1, unsigned long vaddr, unsigned frame,
      short r, short w, short x);
```

Ovu funkciju poziva kod kernela kada treba da postavi ulaze u PMT (po potrebi, oba nivoa) kada je alocirao okvir za neku stranicu virtuelne memorije prilikom obrade straničnе greke. Značenje argumenata je sledeće:
`pmt1`: pokazivač na PMT prvog nivoa (već alocirana prilikom kreiranja procesa);
`vaddr`: puna virtuelna adresa koja se obrađuje (pripada stranici koja je učitana);
`frame`: broj okvira koji je već alociran za datu stranicu;
`r`, `w`, `x`: prava pristupa koja treba postaviti (0 ili 1; čitanje, upis, izvršavanje, respektivno).
Veličine tipova su sledeće: `int` – 32 bita, `long` – 64 bita, `short` – 16 bita.
Na raspolaganju je interna funkcija kernela `alloc_pmt()` koja alocira jednu PMT u memoriji, popunjava sve njene ulaze nulama i vraća broj okvira u kom počinje ta alocirana PMT. Ignorisati sve eventualne greške.

Rešenje: