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: Prvi, februar 2026.
Datum: 13. 3. 2026.

Prvi kolokvijum iz Operativnih sistema 2

Kandidat: _____

Broj indeksa: _____ *E-mail:*_____

Kolokvijum traje 90 minuta. Dozvoljeno je korišćenje literature.

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

Ukupno: _____/30 = _____%

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) Raspoređivanje procesa

U nekom ugrađenom (*embedded*) simetričnom multiprocesorskom sistemu postoji `NUM_OF_PROCESSORS` hardverski potpuno identičnih procesora. Svi procesori izvršavaju isti kod kernela, pristupaju istim zajedničkim strukturama kernela i izvršavaju isti skup procesa. Procesori su numerisani brojevima `0.. NUM_OF_PROCESSORS-1`. Tekući proces koji izvršava dati procesor vraća statička funkcija `Running::get`.

Kernel raspoređuje procese po prioritetu; prioritet procesa vraća funkcija `PCB::getPri` (veća numerička vrednost označava viši prioritet). Prema tome, procesori uvek izvršavaju prvih `NUM_OF_PROCESSORS` trenutno izvršivih procesa najvišeg prioriteta (uvek ih ima barem toliko, jer ima toliko *idle* procesa). Većina procesa su periodični, što znači da se uspavljuju do određenog trenutka nakon svake periodične aktivacije. Podsistem `Timing`, koji brine o merenju vremena, vodi evidenciju o intervalima na koje su uspavani procesi i na onaj koji najskorije ističe podešava hardverski tajmer koji generiše prekid. Kada taj interval istekne i hardverski tajmer generiše prekid, aktivira se proces koji je na dati interval uspavan; ukoliko je potrebno prema raspoređivanju po prioritetima, tada treba uraditi preotimanje procesora.

Prekid obrađuje onaj procesor na koji je trenutno usmeren prekid od hardverskog tajmera. Ovo usmeravanje obavlja poseban hardverski uređaj, kontroler prekida, prostim demultipleksiranjem signala od tajmera na ulaz za prekid procesora na koji je podešen. Podešavanje ovog usmeravanja, upisom broja procesora u odgovarajući registar kontrolera prekida koji zadaje ulaz demultiplekseru, obavlja funkcija `IntCntrl::setIntHndlProc`.

Implementirati funkciju `Timing::setIntHndlProc` koju poziva kernel svaki put kada završi promenu konteksta, tj. nakon promene skupa procesa koji se trenutno izvršavaju; ona treba da preusmeri prekid od tajmera na odgovarajući procesor koji treba da obradi taj prekid i eventualno, ako je potrebno, uradi svoje preotimanje zbog toga što se istekom intervala pojavio nov izvršiv proces.

```
PCB* Running::get(int processor);
Priority PCB::getPri() const;
void IntCntrl::setIntHndlProc(int processor);

void Timing::setIntHndlProc ();
```

Rešenje:

2. (10 poena) Međuprocesna komunikacija pomoću deljene promenljive

Dole je data implementacija ograničenog bafera korišćenjem monitora i klasičnih uslovnih promenljivih na jeziku Concurrent Pascal. Proizvođači u bafer stavljaju zahteve tipa `Request` (^ označava pokazivač) koje potrošači uzimaju i obrađuju, s tim što se proizvođač suspenduje dok njegov zahtev ne bude uzet na obradu. Precizno objasniti koji problem postoji u ovoj implementaciji i pod kojim uslovima se taj problem ispoljava.

```
monitor BoundedBuffer;
  export append, take;

  var
    buf : array[0..size-1] of ^Request;
    top, base : 0..size-1;
    numberInBuffer : integer;
    spaceAvailable, itemAvailable : condition;
    taken : array[0..size-1] of condition;

  procedure append (r : ^Request);
    var oldTop : 0..size-1;
  begin
    while numberInBuffer = size do
      wait(spaceAvailable);
    end while;
    oldTop :=top;
    buf[top] := r;
    numberInBuffer := numberInBuffer+1;
    top := (top+1) mod size;
    signal(itemAvailable);
    wait(taken[oldTop]);
  end append;

  procedure take (var r : ^Request);
  begin
    while numberInBuffer = 0 do
      wait(itemAvailable);
    end while;
    r := buf[base];
    base := (base+1) mod size;
    numberInBuffer := numberInBuffer-1;
    signal(taken[base]);
    signal(spaceAvailable);
  end take;

begin
  numberInBuffer := 0;
  top := 0; base := 0
end;
```

Rešenje:

3. (10 poena) Međuprocesna komunikacija razmenom poruka

Dat je deo koda serverske i klijentske strane. Klijent treba da pošalje sadržaj tekstualnog fajla serveru, a da server sačuva sadržaj tog fajla sa istim imenom u tekućem direktorijumu. Dati kod nije u potpunosti ispravan. Precizno navesti šta treba da se ispravi ili doda da bi kod bio ispravan. Kod koristi klasu `Service` koja je data na vežbama.

```
class ClientUpload {
    private static final int BUFFER_SIZE = 5;
    private Service service;
    ...
    public void uploadFile(File file) throws IOException {
        BufferedReader reader = new BufferedReader(new FileReader(file));

        service.sendMsg("UPLOAD_START");
        service.sendMsg(file.getName());

        List<String> lines = new ArrayList<>();
        String line;
        while ((line = reader.readLine()) != null) {
            lines.add(line);
            for (int i = 0; i < BUFFER_SIZE; i++) {
                line = reader.readLine();
                if (line == null) {
                    break;
                }
                lines.add(line);
            }

            service.sendMsg("UPLOAD#" + lines.size());
            for (String l : lines) {
                service.sendMsg(l);
            }
        }

        reader.close();
    }
}

class ServerUpload {
    private Service service;

    public void handleTasks() throws Exception {
        while (true) {
            String msg = service.receiveMsg();
            if (msg.equals("UPLOAD_START")) {
                handleUpload();
            }
        }
    }

    private void handleUpload() throws Exception {
        File file = new File("new_file.txt");
        PrintStream out = new PrintStream(file);

        while (true) {
            String lineCountMsg = service.receiveMsg();
            if (lineCountMsg == null) {
                break;
            }
```

```
            String[] parts = lineCountMsg.split("#");
            if (parts.length != 2 || !parts[0].equals("UPLOAD")) {
                break;
            }
            int lineCount = Integer.parseInt(parts[1]);
            for (int i = 0; i < lineCount; i++) {
                String line = service.receiveMsg();
                if (line == null) {
                    break;
                }
                out.println(line);
            }
        }
        out.close();
    }
}
```

Rešenje: