Elektrotehnički fakultet u Beogradu
Katedra za računarsku tehniku i informatiku

Predmet: Operativni sistemi 2 (13E113OS2, 13S113OS2)
Nastavnik: prof. dr Dragan Milićev
Školska godina: 2018/2019. (Zadatak važi počev od januarskog roka 2019.)

Projekat za domaći rad
- Projektni zadatak –

Verzija dokumenta: 1.0

Važne napomene: Pre čitanja ovog teksta, **obavezno** pročitati opšta pravila predmeta i pravila vezana za izradu domaćih zadataka! Pročitati potom ovaj tekst **u celini i pažljivo**, pre započinjanja realizacije ili traženja pomoći. Ukoliko u zadatku nešto nije dovoljno precizno definisano ili su postavljeni kontradiktorni zahtevi, student treba da uvede razumne pretpostavke, da ih temeljno obrazloži i da nastavi da izgrađuje preostali deo svog rešenja na temeljima uvedenih pretpostavki. Zahtevi su namerno nedovoljno detaljni, jer se od studenata očekuje kreativnost i profesionalni pristup u rešavanju praktičnih problema!

Uvod

Cilj ovog zadatka jeste implementacija raspoređivača procesa (engl. *scheduler*) za operativni sistem. Raspoređivač treba da obezbedi smeštanje procesa u skup spremnih procesa i odabir narednog procesa za izvršavanje. Operativni sistem ima podršku za izvršavanje procesa na jednom ili više procesora, shodno tome raspoređivač treba da obezbedi podršku za odabir procesa za izvršavanje na bilo kom procesoru.

Zadatak se sastoji od tri dela. Svaki deo definiše jedan algoritam koji raspoređivač treba da implementira. Svaki algoritam nosi po 10 poena. Za uspešnu odbranu projektnog zadatka student mora da uradi bar dva dela, po slobodnom izboru.

Opšti zahtevi

Odnos projekta i korisničke aplikacije

Tražene podsisteme treba realizovati na jeziku Java. Korisničku aplikaciju, koja sadrži test primere, treba povezati sa prevedenim kodom projekta u jedinstven konzolni program. U datoj aplikaciji biće prisutna i funkcija *main*.

Odnos projekta i operativnog sistema domaćina

Projekat se realizuje pod operativnim sistemom Windows 7 x64 ili novijim, koji je u ovom slučaju operativni sistem domaćin. Izradom projekta se ni na koji način ne sme ugroziti ispravno funkcionisanje operativnog sistema domaćina. Svaki eventualni problem koji se pojavi po pokretanju projekta biće smatran kao greška pri izradi projekta. Po završetku rada, okruženje je neophodno ostaviti u neizmenjenom stanju u odnosu na trenutak pre pokretanja projekta, osim onih delova koji se namerno menjaju samim test primerom koji treba da proveri ispravnost projekta. Svi resursi u sistemu koji će biti korišćeni pri izradi projekta moraju biti korišćeni kroz odgovarajući API operativnog sistema domaćina koji je za to namenjen. Deo koda koji je obezbeđen u okviru postavke projekta je pažljivo napisan, i ukoliko se koristi u skladu sa uputstvom za rad, ne može prouzrokovati nikakve probleme i greške pri izvršavanju.

Raspoređivač

Uvod

Potrebno je realizovati tri algoritama za raspoređivanje procesa. Algoritam za raspoređivanje se bira prilikom pokretanja programa i isti algoritam se koristi tokom celokupnog izvršavanja. Sistem može imati više procesora. Rapoređivač sadrži skup svih spremnih procesa u sistemu i vrši raspoređivanje za sve procesore.

Raspoređivač

Opis zadatka i funkcionalnosti

Potrebno je realizovati klasu `Scheduler`, koja predstavlja interfejs raspoređivača. Konkretan algoritam raspoređivanja treba implementirati u klasi koja je izvedena iz klase `Scheduler`.

Klasa `Scheduler` treba da realizuje sledeće funkcionalnosti:

- Smeštanje spremnog procesa u skup spremnih;
- Dohvatanje jednog spremnog procesa za izvršavanje na procesoru čiji je jedinstveni identifikator prosleđen kao parametar poziva. Ukoliko ne postoji nijedan spreman proces, raspoređivač treba da vrati vrednost `null`;
- Statičku metodu za kreiranje raspoređivača sa konkretnim algoritmom raspoređivanja. Parametri metode koji se prosleđuju u nizu stringova, predstavljaju parametre za kreiranje raspoređivača sa konkretnim algoritmom (npr. ime algoritma). Ti parametri se prosleđuju prilikom pokretanja test programa iz komande linije. Format tih parametara se ostavlja u nadležnosti samog rešenja.

Scheduler na jeziku Java

Interfejs klase `Scheduler` je dat u nastavku:

```
public abstract class Scheduler {
    public abstract Pcb get(int cpuId);

    public abstract void put(Pcb pcb);

    public static Scheduler createScheduler(String[] args) {
        //TODO: Implement this method
    }
}
```

Pcb

Opis funkcionalnosti

Klasa `Pcb` je data i objekat klase sadrži sve informacije o jednom procesu. Prilikom smeštanja procesa u skup spremnih smatrati da će ispravno biti postavljeni svi parametri važni za raspoređivanje (npr. prethodno stanje procesa, vreme poslednjeg izvršavanja na procesoru, itd.). Za potrebe realizacije algoritma raspoređivanja, u klasi `Pcb` postoji referenca na objekat klase `PcbData`. Preko tog objekta, raspoređivač može pamtiti podatke koji su od interesa za algoritam raspoređivanja. Tim podacima pristupa samo raspoređivač i niko drugi. Referenca na taj objekat je inicijalno postavljena na vrednost `null`. Interfejs i implementacija klase `PcbData` se ostavljaju u nadležnosti samog rešenja. Klasa `Pcb` ima nabrojivi tip `ProcessState`, koji opisuje moguća stanja u kom proces može da se nalazi. U trenutku kada se proces smešta u skup spremnih, proces je sigurno u stanju spreman, dok postoji mogućnost dohvatanja prethodnog stanja u kom je proces bio neposredno pre nego što je postao spreman. Klasa `Pcb` sadrži globalno dostupan statički niz referenci na procese koji se trenutno izvršavaju. Veličina niza je jednaka broju procesora u sistemu. Jedinstveni identifikator procesora je indeks u tom nizu za dohvatanje procesa koji se trenutno izvršava. Raspoređivač ne trebe da menja sadržaj niza, to se radi u kodu koji poziva usluge raspoređivača.

Klasa `Pcb` realizuje sledeće funkcionalnosti:

- Operaciju za preuzimanje procesa. Proces će nakon te operacije u pogodnom trenutku sačuvati svoj kontekst i osloboditi procesor;
- Dohvatanje jedinstvenog identifikatora procesa;
- Dohvatanje prioriteta procesa koji je postavljen prilikom kreiranja procesa;
- Postavljanje vremenskog kvanta za dužinu izvršavanja na procesoru;
- Dohvatanje vremena koje je proces proveo u najskorijem izvršavanju na procesoru;
- Dohvatanje stanja u kom je proces bio neposredno pre nego što je postao spreman;
- Postavljanje i dohvatanje reference na objekat klase `PcbData`;
- Dohvatanje trenutnog vremena u sistemu;
- Dohvatanje ukupnog broja procesa u sistemu.

Pcb na jeziku Java

Deo interfejsa klase `Pcb` je dat u nastavku:

```
public class Pcb {

    public static Pcb[] RUNNING;    public enum ProcessState {

            RUNNING, READY, BLOCKED, CREATED; }
```

```
    public void preempt();

    public int getId();

    public int getPriority();

    public void setTimeslice(long timeslice);

    public long getExecutionTime();

    public ProcessState getPreviousState();

    public PcbData getPcbData();

    public void setPcbData(PcbData pcbData);

    public static long getCurrentTime();

    public static int getProcessCount();

}
```

Algoritmi za raspoređivanje

Potrebno je realizovati sledeća tri algoritma raspoređivanja: aproksimaciju Shortest-Job-First algoritma, Multilevel Feedback-Queue Scheduling algoritam i uprošćenu verziju Completely Fair Scheduler algoritma. Izbeći koliko je moguće izgladnjivanje u bilo kom algoritmu.

Aproksimacija Shortest-Job-First algoritma

Algoritam može biti sa preuzimanjem ili bez preuzimanja. Procena narednog vremena izvršavanja se vrši eksponencijalnim usrednjavanjem. Vremenski kvant za izvršavanje je 0 (neograničen). Koeficijent za eksponencijalno usrednjavanje i varijantu algoritma je moguće postaviti prilikom kreiranja raspoređivača.

Multilevel Feedback-Queue Scheduling

Raspoređivač ima nekoliko redova spremnih procesa razvrstanih po prioritetu. Prilikom prvog ubacivanja procesa (nakon kreiranja), proces se smešta u red prema prioritetu koji mu je dodeljen prilikom kreiranja. Ukoliko je proces prethodno bio blokiran, smešta se u red većeg prioriteta, a ukoliko mu je isteklo procesorsko vreme smešta se u red nižeg prioriteta. Svaki red ima svoj vremenski kvant koji se dodeljuje procesu prilikom sledećeg izvršavanja. Omogućiti da se broj redova i vremenski kvant za svaki red prosleđuju kao parametri prilikom kreiranja raspoređivača.

Uprošćenu verziju Completely Fair Scheduler algoritma

Algoritam odabira sledeći proces za izvršavanje, koji je od početka svog tekućeg naleta izvršavanja (tj. od kako je došao u red spremnih npr. iz stanja suspenzije) imao najkraće ukupno vreme izvršavanja tokom tekućeg naleta izvršavanja (engl. CPU burst). Vremesnki kvant za sledeće izvršavanje jednak je vremenu koje je proces proveo u skupu spremnih čekajući da dobije procesor podeljen sa brojem procesa.

Testovi

Javni testovi

Javni test-program služi da pomogne studentima da elementarno testiraju svoj projekat. Ovi testovi neće obavezno pokriti sve funkcionalnosti koje projekat treba da ima, ali će testirati većinu tih funkcionalnosti. Da bi se projekat uopšte odbranio, neophodno je da projekat sa javnim testom radi u potpunosti ispravno. Studentima se preporučuje da pored javnog testa naprave i svoje iscrpne testove koji će im pomoći da što bolje istestiraju svoj projekat.

Tajni testovi

Tajni testovi detaljnije testiraju sve zahtevane funkcionalnosti u različitim regularnim i neregularnim situacijama (greške u pozivu ili radu), i nisu unapred dostupni studentima.

Testovi performansi

Testovi performansi mere srednje vreme odziva procesa i ukupno vreme izvršavanja procesa. Ovi testovi nisu obavezni, i mogu, ali ne moraju, doneti dodatne bodove u predroku posle nastave za do 20 najboljih odbranjenih radova. Za potrebe povećanja performansi, student može obezbediti svoj algoritam za raspoređivanje, podesiti parametre već datih algoritama, itd. Kompleksnost algoritma se uzima u obzir prilikom ocenjivanja rada. Težiti da proces dobije isti procesor na kom se prethodno već izvršavao, kako bi se smanjilo kašnjenje zbog promašaja u procesorskom kešu koje bi postojalo kada bi se on izvršavao na drugom procesoru. Voditi računa o raspodeli opterećenja procesora (engl. *load balancing*).

Zaključak

Potrebno je realizovati opisane podsisteme prema datim zahtevima na jeziku Java. Kao integrisano okruženje za razvoj programa (engl. integrated development environment, IDE) moguće je koristiti bilo koje, koje je dostupno u laboratorijama 25, 26, 60 i 70. Testiranje se vrši u laboratorijama katedre na računarima pod operativnim sistemom Windows 10 x64.

Pravila za predaju projekta

Projekat se predaje isključivo kao jedna zip arhiva. U arhivu smestiti samo .java fajlove koji su rezultat izrade projekta. Opisani sadržaj ujedno treba da bude i jedini sadržaj arhive (arhiva ne sme sadržati ni izvršne fajlove, ni biblioteke, ni bilo kakve testove, niti bilo šta što iznad nije opisano). Projekat je moguće predati više puta, ali do trenutka koji će preko imejl liste biti objavljen za svaki ispitni rok i koji će uvek biti pre ispita. Na serveru uvek ostaje samo poslednja predata verzija i ona će se koristiti na odbrani. Za izlazak na ispit neophodno je predati projekat (prijava ispita i položeni kolokvijumi su takođe preduslovi za izlazak na ispit). Nakon isteka roka za predaju, projektni zadaci se brišu sa servera, pa je u slučaju ponovnog izlaska na ispit potrebno ponovo postaviti ažurnu verziju projektnog zadataka.

Sajt za predaju projekta je https://rti.etf.bg.ac.rs/domaci/index.php?servis=os2_projekat

Zapisnik revizija

Ovaj zapisnik sadrži spisak izmena i dopuna ovog dokumenta po verzijama.

Verzija 1.0

Strana	Izmena