Elektrotehnički fakultet u Beogradu
Katedra za računarsku tehniku i informatiku

Predmet: Operativni sistemi 1 (13E112OS1, 13S112OS1)
Nastavnik: prof. dr Dragan Milićev
Asistenti: Živojin Šuštran, Milana Prodanov,
Uroš Radenković, Predrag Obradović
Datum: 22.07.2020.

Modifikacija projektnog zadatka

Važne napomene: Pre početka izrade zadatka pročitati i upoznati se sa samim tekstom **u celini i pažljivo**. Ukoliko u zadatku nešto nije dovoljno precizno navedeno ili student naiđe na neki problem, uvesti razumnu pretpostavku i dalje rešenje bazirati na toj pretpostavci, jer se očekuje kreativnost i profesionalni pristup u rešavanju praktičnih problema!

U toku izrade modifikacije pravila su ista kao ispitna pravila fakulteta. Izrada modifikacije traje **120** minuta. Nakon isteka vremena snimiti ovaj falj na L:\ disk. Nepostojanje rezulata rada na L disku povlači oduzimanje poena.

Modifikacija 1 (za projekat od 20 poena)

- Potrebno je izmeniti funkcionalnost klase `Event` na sledeći način:
    - Uvodi se nova nestatička metoda `void Event::register()`, kojom se pozivajuća nit registruje za događaj. Moguće je da više niti bude registrovano. Kada se desi događaj, sve niti se obaveštavaju onim redosledom kojim su bile registrovane.
    - Izmeniti konstruktor klase `Event` tako da nit koja pravi objekat događaja više nije njen vlasnik, odnosno o događaju sada mogu da budu obaveštene samo niti koje su se registrovale gornjom metodom `Event::register`.
- Napisati test primer koji treba da demonstrira rad dodate funkcionalnosti:
    - Potrebno je napraviti jedan objekat događaja na ulazu `9h` (tastatura), kao i `N = 50` istih korisničkih niti, koje se najpre registruju za taj isti događaj, a zatim se ciklički blokiraju na njemu i simuliraju svoj rad uspavljivanjem (na nekom lokalnom semaforu) ili uposlenim čekanjem. Iskoristiti funkciju `rand()` iz zaglavlja `stdlib.h`, koja vraća cele brojeve od `0` do `RAND_MAX`, za uvođenje slučajne dužine uspavljivanja, odnosno uposlenog čekanja. Voditi računa da se prekid od tastature generiše i prilikom pritiskanja i prilikom otpuštanja tastera (a može se iskoristiti i odgovarajući deo koda iz javnog testa).
    - Neposredno pre blokiranja niti na događaju (u metodi **`Event::wait`**), odnosno nakon primanja signala od događaja (u svojoj **`run`** metodi posle poziva `Event::wait`), nit ispisuje odgovarajuću poruku (primer: **`BLOCKED - THREAD ID = 2`**, odnosno **`SIGNALED, THREAD ID = 2`**).

Modifikacija 2 (za projekat od 30 poena)

- Potrebno je izmeniti funkcionalnost klase `Thread` na sledeći način:
    - Izmeniti metodu za lokalno blokiranje signala tako da je moguće blokirati samo određene `SignalHandler` funkcije. Novi potpis metode treba da bude
      `void Thread::blockSignal(SignalId signal, int mask);`
    - Jedinice u parametru `mask` određuju `SignalHandler` funkcije koje NE TREBA blokirati (najniža cifra odgovara prvoj registrovanoj `SignalHandler` funkciji, sledeća cifra odgovara drugoj, itd⋯), dok ostale TREBA blokirati. Parametar `mask` podrazumevano ima sve nule. Za rad sa bitovima po potrebi koristiti aritmetičko šiftovanje (<<, >>), bitsko "`I`" i

"ILI", (&, |), invertovanje bita (~) i heksadecimalne brojeve (0x10).

- SignalHandler funkcije koje se blokiraju sada NE TREBA obrađivati nakon deblokiranja, odnosno treba ih u potpunosti ignorisati.

- Pretpostaviti da će sve SignalHandler funkcije da budu registrovane pre prvog poziva metode Thread::blockSignal.

- Napisati test primer koji treba da demonstrira rad dodate funkcionalnosti

  - Potrebno je jednoj niti poslati bar 100 signala. Nit koja šalje signale blokira se 200 perioda prekida od timer-a nakon svakog petog poslatog signala. Mora biti bar dva različita signala (*različita* od 0, 1 i 2). Takođe, svaki signal treba da ima bar četiri registrovane SignalHandler funkcije. Prilikom slanja signala potrebno je obezbediti odgovarajući ispis (primer: **Signal 3 - sent**). SignalHandler funkcije takođe treba da sadrže odgovarajući ispis (primer: **Signal 3 Signal Handler 1 – handled**, *Napomena*: brojeve signala i SignalHandler funkcija upisati direktno u kod, ali voditi računa da se brojevi SignalHandler funkcija poklope sa njihovim redosledom registracije).

  - Nit koja prima signale se ciklički blokira na svom Event objektu vezanom za ulaz broj 9 (tastatura). Voditi računa da se prekid od tastature generiše i prilikom pritiskanja i prilikom otpuštanja tastera (a može se iskoristiti i odgovarajući deo koda iz javnog testa). Kada dođe prekid od tastature, nit se odblokira i obrađuje sve do tada pristigle, a neobrađene signale. Svaku petu iteraciju nit treba da blokira nula ili više SignalHandler funkcija od jednog ili više signala pozivom izmenjenje metode Thread::blockSignal. Potrebno je unutar te metode dodati i ispis o tome koji su signal hendleri blokirani, odnosno koji nisu (primer: **Thread id = 3, Signal 3, BLOCKED HENDLERS 0, 2, NON-BLOCKED HENDLERS 1, 3,** *Napomena*: redni brojevi SignalHandler funkcija odgovaraju njihovom redosledu registracije).