Sistemi u Realnom Vremenu
Prvi domaći zadatak

Konkurentna mapa

Napisati aplikaciju nad FreeRTOS koja omogućava konkurentno faktorisanje proizvoljnog zadatog broja i smeštanje rezultata faktorizacije u mapi. Proces faktorizacije treba da koristi mapu kako bi algoritam bio nešto efikasniji od obične brute force pretrage.

Faktorizacija brojeva

Faktorizacija podrazumeva rasčlanjivanje zadatog broja na njegove proste činioce. Na primer:

48 = 2 * 2 * 2 * 2 * 3

Brute force pristup ovom problemu bi bio (broj koji faktorišemo je `num`):

- `divisor = 2`
- `while num > 1:`
  - `if num % divisor = 0:`
    - `add_factor(divisor) ;dodaje divisor u niz factors`
    - `num = num / divisor`
  - `else`
    - `divisor = divisor + 1`
- `add_to_map(num, factors)`

Ovaj proces može da se ubrza ako radimo faktorizaciju za neki broj koji može da se razloži na brojeve koje smo već faktorisali. Ako je `factorized` mapa sa rešenjima prethodnih faktorizacija, i `get_factorization` funkcija koja nam daje proste faktore za traženi broj, primenjujemo sledeću optimizaciju:

- `divisor = 2`
- `while num > 1:`
  - `if num % divisor = 0:`
    - `add_factor(divisor) ;dodaje divisor u niz factors`
    - `num = num / divisor`
    - `if num in factorized:`
      - `add_factors(get_factorization(num))`
      - `break`
  - `else`

            - `divisor = divisor + 1`
- `add_to_map(num, factors)`

Mapa faktorisanih brojeva

Treba implementirati mapu koja će omogućiti operacije koje su gore opisane. Mapa treba da zadovoljava sledeće uslove:

- Maksimalni broj elemenata u mapi je fiksiran pomoću globalnog parametra `MAX_MAP_SIZE`.
- Ključevi u mapi su tipa `long`, a vrednosti u mapi su `long*` - pokazivači na dinamički alocirane nizove. Ključ predstavlja broj za koji je urađena faktorizacija, a niz sadrži proste faktore tog broja.
- Za očekivati je da `MAX_MAP_SIZE` bude manje od `LONG_MAX`. Kvalitet funkcije heširanja nije bitan. U slučaju kolizije, sukobljeni elementi treba da se čuvaju u povezanoj listi.
- Operacije:
    - `int iMapPut(long num, long* factors)` - Dodaje stavku u mapu. Pretpostavlja da je niz `factors` dinamički alociran. Vraća:
        - -1 ako je mapa puna, tj. do sada je uspešno obavljeno `MAX_MAP_SIZE` ubacivanja.
        - -2 ako element već postoji u mapi.
        - 1 ako je element uspešno ubačen u mapu.
    - `long* lMapGet(long num, TickType_t timeout)` - Vraća faktore za traženi broj. Ako rezultat ne postoji u mapi, blokira trenutni task dok se rezultat ne pojavi ili istekne zadato vreme. Ako se rezultati ne pojave pre isteka zadatog broja tick-ova, funkcija se završava i vraća `NULL`. Ako parametar `timeout` ima vrednost `portMAX_DELAY`, čeka se neograničeno dugo.
    - `int lMapSize()` - Vraća broj elemenata koji se trenutno nalaze u mapi, tj. broj uspešno obavljenih `put` operacija.

Testiranje rada sistema

Formirati testove koji ilustruju sledeća ponašanja sistema:

1. Serijalizovano izvršavanje - jedan task računa faktore za nekoliko brojeva. Pokušaj faktorisanja istog broja više puta. Pokušaj faktorisanja broja nakon što je mapa došla do granice `MAX_MAP_SIZE`. Proveriti da li su sračunati faktori dobri, i da nema faktora za brojeve za koje nije vršen račun. Proveriti da `get` čeka zadati broj tick-ova ako nema rezultata za traženi broj.
2. Konkurentno izvršavanje - veći broj taskova koji vrše faktorizaciju. Svaki task faktoriše brojeve u nekom opsegu. Pobrinuti se da ima taskova kojima se opsezi preklapaju, tako da je moguće da više taskova pokuša da uradi `put` za istu vrednost

konkurentno. Test treba da ilustruje da sistem ne radi ako nema sinhronizacije, kao i da radi ako ima sinhronizacije.

Rok i bodovanje

Rok za izradu domaćeg zadatka je ponoć, 3-4. novembar.

Domaći zadatak mora da se odbrani da bi bio bodovan. Odbrana podrazumeva odgovaranje na pitanja u vezi izrade domaćeg zadatka, ili izvršavanja neke jednostavne izmene nad zadatkom na licu mesta. Domaći se radi isključivo individualno. Bitno je da svaki student koji brani domaći zadatak brani kod koji je samostalno napisao. U suprotnom će domaći zadatak biti bodovan sa -15 poena.

Termin za odbranu prvog domaćeg zadatka će biti u subotu, 09. novembra. Tačan raspored odbrana će biti objavljen nakon što su svi zadaci predati.

Zadatak se predaje putem emaila na bmilojkovic@raf.rs. Neophodno je arhivirati projekat i poslati ga kao attachment uz email. Format za naziv projekta i arhive treba da bude: `srv_2019_d1_ime_prezime_indeks.` Na primer:

- `srv_2019_d1_student_studentic_rn0101`

Email pored attachment-a ne mora da ima telo poruke, ali treba da sadrži subject formatiran kao: `[SRV 2019] D1 ime_prezime_indeks.` Na primer:

- `[SRV 2019] D1 student_studentic_rn0101`

Prvi domaći zadatak vredi 15 poena i boduje se na sledeći način:

- Faktorizacija mora da se obavlja po algoritmu koji koristi mapu. U suprotnom, zadatak vredi 0 poena.
- Implementacija mape: najviše 10 poena. Moguća bodovanja za ovu stavku su:
  - Implementacija zasnovana na muteksu: 3 poena.
  - Implementacija koristi `MAX_MAP_SIZE` semafora: 5 poena.
  - Implementacija sa konstantnim brojem semafora: 10 poena.
- Pravilno napisan test (važi samo uz implementaciju mape od bar 5 poena): 5 poena.