

## Lab-3. Sinkronizacija monitorom

Korištenjem POSIX sučelja, monitor se ostvaruje preko varijabli zaključavanja (mutex) i redova uvjeta. Ulazak i izlazak iz monitora se ostvaruje sučeljima:

```
int pthread_mutex_lock(pthread_mutex_t *mutex);
int pthread_mutex_unlock(pthread_mutex_t *mutex);
```

gdje je *mutex* globalna varijabla.

Čekanje u redu uvjeta, odnosno, oslobađanje iz reda ostvaruje se sučeljima:

```
int pthread_cond_wait(pthread_cond_t *cond, pthread_mutex_t *mutex);
int pthread_cond_signal(pthread_cond_t *cond);
int pthread_cond_broadcast(pthread_cond_t *cond);
```

gdje su *mutex* i *cond* globalne varijable. Prva funkcija `pthread_cond_wait` smije se pozvati samo unutar monitora, kad je *mutex* već zaključan.

Primjer korištenja tih sučelja prikazan je primjerom u nastavku.

```
#include <stdio.h>
#include <pthread.h>
#include <unistd.h>
#include <time.h>
#define BR_DRETVI 5
#define BROJI_DO 25

pthread_mutex_t m; //monitor
pthread_cond_t red; //red uvjeta
int broj; //1, 2, ...
int uzeo_broj = 0;
time_t zadnja_obrada = 0;
void *dretva (void *p);
void *kada_nitko_nece(void *p);

int main ()
{
    pthread_t t[BR_DRETVI + 1];
    int i, id[BR_DRETVI];

    pthread_mutex_init (&m, NULL);
    pthread_cond_init (&red, NULL);
    broj = 2;
    zadnja_obrada = time(NULL);

    for (i = 0; i < BR_DRETVI; i++) {
        id[i] = i + 2;
        pthread_create (&t[i], NULL, dretva, &id[i]);
    }
    pthread_create (&t[i], NULL, kada_nitko_nece, NULL);
```

```

    for (i = 0; i < BR_DRETVI + 1; i++)
        pthread_join (t[i], NULL);

    pthread_mutex_destroy (&m);
    pthread_cond_destroy (&red);

    return 0;
}

void *dretva (void *p)
{
    int id = *((int *) p);

    pthread_mutex_lock (&m);
    while (broj <= BROJI_DO) {
        while (broj <= BROJI_DO && (uzeo_broj != 0 || broj % id != 0))
            pthread_cond_wait (&red, &m);
        if (broj > BROJI_DO)
            break;
        uzeo_broj = 1;
        pthread_mutex_unlock (&m);

        printf ("Dretva %d radi s brojem %d\n", id, broj);
        sleep(1);
        printf ("Dretva %d staje s radom\n", id);

        pthread_mutex_lock (&m);
        uzeo_broj = 0;
        broj++;
        zadnja_obrada = time(NULL);
        pthread_mutex_unlock (&m);
        pthread_cond_broadcast (&red);
        sleep(BR_DRETVI - (id - 1));
        pthread_mutex_lock (&m);
    }
    pthread_mutex_unlock (&m);
    pthread_cond_broadcast (&red);

    return NULL;
}

void *kada_nitko_nece(void *p)
{
    pthread_mutex_lock (&m);
    while (broj <= BROJI_DO) {
        if (uzeo_broj != 0) {
            //netko je uzeo broj
            pthread_cond_wait (&red, &m);
        }
    }
}

```

```

else if (zadnja_obrada + BR_DRETVI > time(NULL)) {
    //daj im vremena da uzmu broj
    pthread_mutex_unlock (&m);
    sleep(zadnja_obrada + BR_DRETVI - time(NULL));
    pthread_mutex_lock (&m);
}
else {
    printf("Nitko nece broj %d, povecavam ga\n", broj);
    broj++;
    zadnja_obrada = time(NULL);
    pthread_cond_broadcast (&red);
}
}
pthread_mutex_unlock (&m);
}

```

U primjeru se stvaraju dretve koje kao argument dobiju (preko kazaljke) svoj broj (*id*), u rasponu od 2 do 6. Ovisno o trenutno vrijednosti varijable *broj*, ona dretva čiji *id* dijeli broj bez ostatka ulazi u "obradu" tog broja. Naravno, za neke brojeve postoji više takvih dretvi. Npr. broj 12 je djeljiv sa šest, četiri, tri, dva i jedan. Jedna od tih dretvi će to detektirati i ući u obradu. Koja, ovisi o tom trenutku, gdje se koja dretva nalazi u redu pripravnih i slično (može bilo koja od navedenih, ne može se predvidjeti!). U sustavu postoji i dretva koja će detektirati brojeve koje niti jedna stvorena dretva ne može uzeti te tada povećati broj za jedan.

## Zadatak

Nadograditi program iz druge vježbe (ali u lab3) tako da se broj radnih dretvi može dinamički mijenjati. Npr. na signal `SIGUSR1`, osim prijašnje operacije, postaviti i nasumično povećanje ili smanjenje broja dretvi za slučajni broj (1-5).

Neka u programu postoje dvije globalne varijable, npr. `broj_dretvi` te `postavljeni_broj_dretvi`. Prva označava koliko trenutno ima radnih dretvi, a druga koliko ih treba biti. Kada je broj dretvi premali, onda stvoriti još dretvi (u obradi tih signala koje mijenjaju broj dretvi). Kada je broj dretvi prevelik, onda će neke dretve same završiti s radom (na kraju svoje petlje ući u monitor, usporediti te dvije varijable te po potrebi završiti s radom s `pthread_exit(NULL)`, ali prije toga ažurirati `broj_dretvi` i izaći iz monitora).

Do sada korištene semafore zamijeniti monitorima i redovima uvjeta (po potrebi). Npr. u zaključavanjima samo `sem_wait` zamijeniti s `pthread_mutex_lock` (slično i s `post/unlock`).

Promijeniti blokiranje dretvi na praznom međuspremniku tako da se tamo koristiti red uvjeta (`pthread_cond_wait + pthread_cond_broadcast` iz signala).

Obzirom na dinamičko stvaranje i završetak dretvi, nije potrebno pamti opisnike dretvi, može se koristiti i samo jedna varijabla `pthread_t opisnik`; koja će se prepisati pri svakom novom stvaranju dretve. Takve bi dretve također trebalo (ali nije nužno) označiti i kao „odvojive“ pri stvaranju ([primjer](#)) ili kasnije s `pthread_detach(pthread_self())`.

Preporuča se prvo ugraditi monitor i redu uvjeta te tek kad to proradi napraviti ostatak.