

3. Sinkronizacijski mehanizmi

3.1. Semafori u UNIX-u (uz sučelja `sem_*`)

Sučeljem `sem_*` ostvaruje se opći semafor (OSEM s predavanja). Ulazak i izlazak iz kritična odsječka može se ostvariti ovim semaforom uz početnu vrijednost 1.

```
int sem_init(sem_t *sem, int mprocesi, unsigned int koliko );
int sem_post(sem_t *sem); – PostaviOSEM
int sem_wait(sem_t *sem); – ČekajOSEM
int sem_destroy(sem_t *sem); – obriši semafor
```

Koncept korištenja

```
sem_t s; //globalna varijabla

//inicijalizacija, npr. u funkciji main() prije stvaranja dretvi
sem_init(&s, 0, 5); //početna vrijednost = 5

//u novim dretvama
sem_wait(&s); //ČekajOSEM
...
sem_post(&s); //PostaviOSEM

//u funkciji main, nakon završetka ostalih dretvi
sem_destroy(&s);
```

Korištenje za dretve različitih procesa

```
sem_t *sem; //globalna varijabla = za kazaljku na objekt u zajedničkoj memoriji

void proces(int id)
{
    ...
    sem_wait(sem); i/ili sem_post(sem);
    ...
}

int main()
{
    ...
    ID = shmget(IPC_PRIVATE, sizeof(sem_t), 0600);
    sem = shmat(ID, NULL, 0);
    shmctl(ID, IPC_RMID, NULL); //moze odmah ovdje, nakon shmat, ili na kraju nakon
    shmdt

    sem_init(sem, 1, 5); //početna vrijednost = 5, 1=>za procese

    ... fork() ...
    ...
    ... wait(NULL) ...

    sem_destroy(sem);
    shmdt(sem);

    return 0;
}
```

Primjeri na webu (osnovni koncepti OSa: semafor)

3.2. Monitori

Monitor se ostvaruje s pomoću *varijabli međusobnog isključivanja* i *uvjetnih varijabli*

1. varijable međusobnog isključivanja (*mutex*)

- kritični odsječak se zaključa s pomoću kontrolnih varijabli (ključeva)
- "zaključavanje" => ulaz u monitor => `pthread_mutex_lock(&m)`
- "otključavanje" => izlaz iz monitora => `pthread_mutex_unlock(&m)`

2. uvjetne varijable = red uvjeta

- red uvjeta se koristi isključivo unutar monitora!
- korištenje reda uvjeta pri "zauzimanju sredstava":
 - a) dretva najprije ulazi u monitor
 - b) u monitoru provjerava može li zauzeti tražena sredstva, tj. može li nastaviti s radom
 - provjera se izvodi korištenjem varijabli koje definiraju stanje sustava u programu, van jezgre – npr. jesu li lijevi i desni štapić slobodni (kod problema pet filozofa)
 - c) ako ne može nastaviti, dretva poziva `pthread_cond_wait(&red, &m)`
 - funkcija `pthread_cond_wait(&red, &m)` uključuje tri operacije
 - i) izbacuje dretvu iz monitora `m` (i propušta iduću unutra)
 - ii) blokira dretvu i stavlja ju u red uvjeta `red`
 - iii) nakon što je dretva odblokirana iz reda uvjeta, dretvu ponovno stavlja u monitor `m` ili u red za ulaz u monitor (ako je u njemu već neka druga dretva)
 - prve dvije operacije (i, ii) su "atomarne", obavljaju se u jednoj jezgrinoj funkciji
 - treća operacija (iii) obavlja se nakon odblokiranja te dretve
 - blokiranu dretvu u redu uvjeta može odblokirati samo druga dretva
- korištenje reda uvjeta pri "oslobađanju sredstava":
 - a) dretva najprije ulazi u monitor
 - b) u monitoru mijenja zajedničke varijable (opcionalno)
 - c) dretva oslobađa samo prvu ili sve dretvu iz reda uvjeta pozivima:
 `pthread_cond_signal(&red)` ili `pthread_cond_broadcast(&red)`
 - d) dretva izlazi iz monitora (i omogućava drugima ulaz)
 - e) ukoliko dretva pozove jednu od navedenih funkcija, ali u tom trenutku nije bilo blokiranih dretvi, taj poziv neće ništa napraviti – to nije semafor koji bi u takvom slučaju povećao vrijednost semafora – uz red uvjeta se ne veže vrijednost!
 - poziv `pthread_cond_wait(&red, &m)` uvijek blokira pozivajuću dretvu
 - poziv `pthread_cond_signal(&red)` odblokira dretvu ako ima takva dretva u zadanom redu uvjeta, u protivnom ne radi ništa

Varijable međusobnog isključivanja i redovi uvjeta trebaju biti globalne varijable.

Sučelja

```
//globalne varijable
pthread_mutex_t ključ;
pthread_cond_t uvjet;

pthread_mutex_init(&ključ, NULL);
pthread_cond_init(&uvjet, NULL);

pthread_mutex_lock(&ključ);

pthread_cond_wait(&uvjet, &ključ);
pthread_cond_signal(&uvjet);
pthread_cond_broadcast(&uvjet);

pthread_mutex_unlock(&ključ);
```

Koncept korištenja

```
#include <pthread.h>
//globalne varijable
pthread_mutex_t m;
pthread_cond_t red;

//inicijalizacija, npr. u funkciji main() prije stvaranja dretvi
pthread_mutex_init(&m, NULL);
pthread_cond_init(&red, NULL);

//u novim dretvama (jedan ili oba iduća bloka)
//prvi blok
pthread_mutex_lock(&m);           // Uđi_u_monitor(m)
while (uvjet_nije_zadovoljen)
    pthread_cond_wait(&red, &m);  // Čekaj_u_redu_uvjeta(red, m)
... (napravi nešto u monitoru)
pthread_mutex_unlock(&m);         // Izađi_iz_monitora(m)

//drugi blok
pthread_mutex_lock(&m);           // Uđi_u_monitor(m)
... (napravi nešto u monitoru)
pthread_cond_signal(&red);        // Oslobodi_iz_reda_uvjeta(red)
pthread_mutex_unlock(&m);         // Izađi_iz_monitora(m)

//u funkciji main, nakon završetka ostalih dretvi
pthread_mutex_destroy(&m);
pthread_cond_destroy(&red);
```

Isječak kôda 3.1. Primjer s ping-pong dretvama

```
#include <stdio.h>
#include <pthread.h>
#include <time.h>
#include <stdlib.h>
#include <unistd.h>

pthread_mutex_t m; //monitor
pthread_cond_t red[2]; //redovi uvjeta

char *ispis[] = {"ping", "pong"};
int na_redu = 0;
int radi = 0;
void *dretva(void *p);

int main() {
    pthread_t t;
    void *x[] = {NULL, (void*)1};

    pthread_mutex_init(&m, NULL);
    pthread_cond_init(&red[0], NULL);
    pthread_cond_init(&red[1], NULL);

    while(1) {
        pthread_create(&t, NULL, dretva, x[rand()%2]);
        sleep(1);
    }
    return 0;
}

void *dretva(void *p) {
    int tip = 0;
    if (p)
        tip = 1;

    pthread_mutex_lock(&m);
    printf("Kreće dretva %s\n", ispis[tip]);
    while(na_redu != tip || radi)
        pthread_cond_wait(&red[tip], &m);
    radi = 1;
    pthread_mutex_unlock(&m);

    printf("%s\n", ispis[tip]);

    pthread_mutex_lock(&m);
    radi = 0;
    na_redu = 1 - tip;
    pthread_cond_signal(&red[1-tip]);
    pthread_mutex_unlock(&m);

    return NULL;
}
```

Još primjera na webu: <https://github.com/ljelenkovic/os-fer-primjeri/tree/master/monitori>