Curs 10 – Sisteme de operare

- thread
- mutex
- rwlock
- condvar
- semafor
- barieră

→ două threaduri care își incrementează alternativ

```
round.c
#include <stdio.h>
#include <pthread.h>
int n = 0;
pthread_mutex_t mtx = PTHREAD_MUTEX_INITIALIZER;
void *f(void *a) { while (n<10) { printf("%d\n", n++); }
    return NULL;
}
int main(int argc, char **argv) {
    pthread_t ta, tb;
    pthread_create(&ta, NULL, f, "A");
    pthread_create(&tb, NULL, f, "B");
    pthread_join(ta, NULL);
    pthread_join(tb, NULL);
    return 0;
}
```

```
int id = ((char*)a)[0] - 'A';
while (n<10) {
    pthread_mutex_lock(&mtx);
    if (n<10) {
        if (n%2 == id) {
            printf("%s: %d\n", (char*)a, n++);
        }
    } else {
        pthread_mutex_unlock(&mtx);
        break;
    }
    pthread_mutex_unlock(&mtx);
}
return NULL;
```

```
void *fa(void *a) {
    while (1) {
        pthread_mutex_lock(&ma);
        if (n<10) {
            printf("A: %d\n", n++);
        } else { pthread_mutex_unlock(&mb); break; }
        pthread_mutex_unlock(&mb);
    }
    return NULL;
}

pthread_t ta, tb;
pthread_mutex_lock(&mb);
```

lock(&mb)

→ "B"

unlock(&ma)

→ putem face cu semafoare

buffer de n poziții

- T_p → pun date în buffer
- T_c → scot date din buffer

P(0) → plin
G(N) → gol

wait()
post()

Tp	Tc
sem_wait(G)	sem_wait(P)
pune date	scoate date
sem_post(P)	sem_post(G)

trylock

→ trambulină : 2 semafoare → nr de copii
→ nr de kg

trampoline {

```
sem_t
~~int~~ kg = 200;
~~int~~ persons = 4;
sem_t
sem_init (&kg, 0, 200);
sem_init (&persons, 0, 4);
for (i=0; i<100; i++){
    pthread_create (&t[i], NULL, f, NULL);
}
for (i=0; i<100; i++) {
    pthread_join( t[i], NULL);
}
sem_destroy (&kg);
sem_destroy( &persons);
```

```
void* f ( void* a){
    int w = rand() % 100;
    ~~sem(wait)~~ sem_wait(&persons);
    for (i=0; i<w; i++){
        sem_wait (&kg);
    }
    // jump
    printf(" %d are %d kg și sare \n",
        pthread_self(), w);
    for (i=0; i<w; i++){
        sem_post (&kg);
    }
    sem_post (&pers);
    return NULL;
```