Probleme threads:

* (r-w) Vinderea de bilete pe un site unde exista cumparatori si oameni care doar se uita
* (var cond) o cantina unde se termina mancarea. Bucatarul sta linistit doarme si trebuie sa se trezeasca sa aduca mancarea apo isa se culce la loc. Pana se trezeste mancarea poate aparea cumvaa.
* (bariere) copii pleaca in excursie, fiecar emasina are n locuri, masinile pleaca cand sunt pline
* (semaf) pt zone de cod incarcate

**Var conditionala**

* Functii (**pthread\_cond\_)**
  + Init
  + Wait
  + Signal (o apeleaza cel care vrea sa trezeasca un thread care asteapta)
  + Broadcast (ii trezeste pe toti nu doar unul ca signal)
  + Destroy

|  |  |
| --- | --- |
| asteapta | trezeste |
| lock(&m)  while(!conditie){  wait(&c, &m);  (in interiorul fct wait:  unlock(&m)  asteapta semnalarea  lock(&m)  )  }  unlock(&m) | lock(&m)  if(conditie){  signal(&c);  }  unlock(&m); |

**Bariere**

* Functii (**pthread\_barrier\_)**
  + Init
  + Wait (niciun thr nu trece mai departe pana nu ua terminat toate)
  + Destroy

**Semafoare**

* Functii (**pthread\_sem\_)**
  + Init (initializarea cu o anumita valoare, dc e initializat cu 1 se numeste semafor binar si se comporta ca un mutex)
  + Wait (decrementeaza valoarea pana cand ajunge < 0 si asteapta pana cand unul din thr apeleaza post)
  + Post (incrementeaza valoarea)
  + Destroy
* **Suma numere cu 10 threaduri**

void \* f(void \* a){

int i, sum = 0;

int k = \*(int\*) a;

for(i=k; i<n; i+=10)

{

sum+=arr[i];

}

pthread\_mutex\_unlock(&m);

total+=sum;

pthread\_mutex\_lock(&m);

* **RW lock**

struct rwl\_t{

pthread\_mutex\_t m;

int r = 0; //nr de readeri

pthread\_cond\_t c;

}

void rwl\_rdlock(struct rwl\_t \*l){

pthread\_mutex\_lock(&l->m);

l->r++;

pthread\_mutex\_unlock(&l->m);

}

void rwl\_rdunlock(struct rwl\_t \*l){

pthread\_mutex\_lock(&l->m);

l->r--;

if(l->r==0){

pthread\_cond\_broadcast(&l->c);

}

pthread\_mutex\_unlock(&l->m);

}

void rwl\_wrlock(struct rwl\_t \*l){

pthread\_mutex\_lock(&l->m);

while(l->r>0){

pthread\_cond\_wait(&l->c, &l->m);

}

}

void rwl\_wrunlocck(struct rwl\_t \*l){

pthread\_mutex\_unlock(&l->m);

}

* **Bariera**

struct bar\_t{

int n;

int k = 0;

pthread\_cond\_t c;

pthread\_mutex\_t m;}

void bar\_wait(struct bar\_t \*b){

pthread\_mutex\_lock(&b->m);

b->k++;

while(b->k % b->n == 0) {

pthread\_cond\_wait(&b->c, &b->m);

}

pthread\_cond\_signal(&b->c);

pthread\_mutex\_unlock(&b->m);

}

* **Problema albine miere padurari**

#include <stdio.h>

#include <pthread.h>

#define URSI 5

#define ALBINE 10

int miere = 0;

pthread\_mutex\_t m;

pthread\_cond\_t c;

pthread\_barrier\_t b;

pthread\_sem\_t s;

void \*albina(void \*a){

pthread\_barrie\_wait(&b);

while(1){

pthread\_mutex\_lock(&m);

miere++;

pthread\_mutex\_unlock(&m);

}

return null;

}

void\* urs(void \* a){

pthread\_barrier\_wait(&b);

while(1){

sem\_wait(&s);

pthread\_mutex\_lock(&m);

if(miere>=10)

miere-=10;

else{

pthread\_cond\_signal(&c); //nu se intampla instant

}

pthread\_mutex\_unlock(&m);

sem\_post(&s);

}

return null;

}

void\* padurar(void \* a){

pthread\_barrier\_wait(&b);

while(1){

pthread\_mutex\_lock(&m);

while(miere>=10)

pthread\_cond\_wait(&c, &m);

miere+=100;

pthread\_mutex\_unlock(&m);

}

return null;

}

int main(int argc, char \*\* argv){

pthread\_t ursi[URSI], albine[ALBINE], pad;

int I;

pthread\_mutex\_init(&m, NULL);

pthread\_cond\_init(&c, NULL);

pthread\_barrier\_init(&b, NULL, URSI+ALBINE+1);

sem\_init(&s, 0, 2);

for(int I = 0; i<ALBINE; i++)

pthread\_create(&albine[i], NULL, albina, NULL);

for(int I = 0; i<URSI; i++)

pthread\_create(&ursi[i], NULL, urs, NULL);

pthread\_create(&pad, NULL, padurar, NULL);

for(int I = 0; i<ALBINE; i++)

pthread\_join(&albine[i], NULL);

for(int I = 0; i<URSI; i++)

pthread\_join(&ursi[i], NULL);

pthread\_join(&padurar, NULL);

pthread\_mutex\_destroy(&m);

pthread\_cond\_destroy(&c);

pthread\_barrier\_destroy(&b);

sem\_destroy(&s);

return 0;

}