Programarea Aplicaţiilor în Timp Real
Sem. 2, 2022-2023

Problema prioritatii pacientilor

- Problema de sincronizare -

Nume Prenume	Grupa	Adrese de e-mail
Stoian Alexandru-Gabriel	333AB	alexandru.stoian01@stud.acs.upb.ro
Marin Florin Daniel	333AB	florin_daniel.marin@stud.acs.upb.ro
Stamule Theo	333AB	theo.stamule@stud.acs.upb.ro

Contents

1 Introducere. Definire problemă

La o unitate de urgenta care accepta 10 pacienti pe zi, vin aleatoriu trei tipuri de pacienti la momente din zi aleatorii: cu prioritate medicala grava, cu prioritate medicala medie, cu prioritate medicala redusa. Conducerea spitalului a hotarat ca pacientii sa intre la consultatie in functie de prioritatea nevoii medicale.

Problema se rezolva in real time programming unde 1s = 10 min din viata reala folosind operatii de sincronizare.

Definire problemă

- Task G - Preluare in cabinet pacient cu prioritate medicala **Grava**
- Task M - Preluare in cabinet pacient cu prioritate medicala **Medie**
- Task R - Preluare in cabinet pacient cu prioritate medicala **Redusa**
- Task addQueue - Populare coada de asteptare cu pacienti in stare Grava , Medie si Redusa la momente aleatorii de timp
- Task monitorizare - Decide ce pacient va intra in cabinet cand acesta se elibereaza

Constrangeri

1. Maxim 10 de pacienti in sala de asteptare
2. Pacientul cu prioritate grava intra in cabinet daca cabinetul este gol
3. Pacientul cu prioritate medie intra in cabinet daca:
   - cabinetul este gol
   - nu exista pacient cu prioritate grava in coada de asteptare
4. Pacientul cu prioritate redus intra in cabinet daca:
   - cabinetul este gol
   - nu exista pacient cu prioritate grava in coada de asteptare
   - nu exista pacient cu prioritate medie in coada de asteptare
5. Timpul de consultare al pacientului cu prioritate grava este intre 60-120 min
6. Timpul de consultare al pacientului cu prioritate medie este intre 30-60 min
7. Timpul de consultare al pacientului cu prioritate redusa este intre 10-30 min

2 Analiza problemei

Solutia problemei pe larg

Noi ne-am gandit la implementarea unei solutii care nu inregistreaza datele pacientilor intr-o structura reala, iar coada de asteptare (sala) definita ulterior va fi o coada abstracta, pur teoretica. Toate datele despre coada sunt cunoscute din valoriile unor semafoare. Vom defini cate-un semafor pentru fiecare tip de pacient, un semafor pentru semnalarea locurilor libere si ocupate din coada virtuala. Problema nu include constrangeri despre identificarea pacientilor sau despre intrarea lor in cabinet dupa timpul in care au ajuns in sala, deci cerinta problemei ramane doar accesul corect la cabinet a tipurilor de pacienti dupa prioritatea lor, identitatea lor nefiind importanta pentru problema data, ci doar tipul de prioritate medicala: grava, medie sau redusa.

Dificultatea problemei sta in atribuirea corecta a resursei critice, care este accesul in cabinet, pacientilor cu prioritatea mare, indiferent de timpul in care au ajuns in coada de asteptare. Prioritizarea pacientilor este, deci, nucleul de interes pentru acest tip de problema, deci o solutie pentru sincronizarea pacientilor trebuie gasita si utilizata. Solutia noastra pentru sincronizare este de a cauta si a decide intr-un timp de maxim 5 minute tipul pacientului care va intra in cabinet, in momentul in care carbinetul se deschide.

Solutia problemei este: politetea pacientilor, prin asteptarea pacientilor cu nevoi medicale mai mari decat ei sa ocupe cabinetul inaintea lor. Pacientii cu prioritate grava intra in cabinet in momentul in care acesta se deschide. Pacientii cu prioritate medie asteapta aprox. 2 minute ca cei dintai sa intre. La expirarea timpului de politete, intra ei in cabinet. In sfarsit, pacientii cu prioritatea redusa, asteapta 4 minute, iar daca altii cu nevoie mai mare decat ei nu au intrat, atunci vor intra ei in cabinet. In program, acest algoritm poarta numele de Monitorizare, iar in descrierea solutiei de mai sus, procedeul poate fi realizat de constiinta fiecarui pacient, sau mai realistic vorbind, de un medic sau asistent, care de fiecare data cand un pacient iese din cabinet, cauta si alege din sala pe cei cu nevoie medicala urgenta, in favoarea celor ce asteapta de mai mult timp la coada.

Am identificat o posibila solutie si am realizat o paralela intre situatiile reale posibile si programul gandit de noi spre rezolvarea problemei. Cu siguranta exista metode mai eficiente d.p.d.v al timpului; suntem curiosi si deschisi spre orice metoda mai eficienta prin metode Real Time Programming. In continuare, numim aceasta solutie de sincronizare: "Politetea pacientilor".

Problema se aseamana problemei tipice Producator-Consumator, unde producatorul este task-ul addQueue, care adauga in coada de asteptare pacienti, consumatorul este task-ul monitorizare, care decide ce pacient are acces la resursa critica - cabinetul, iar buffer-ul sau queue este coada de asteptare care nu exista prin sine, nefiind creata, si exista prin existenta task-urilor G, M si R. Coada - G, M, R sunt "shared resource" intre producator-pacienti si consumer-cabinet. Aceasta paralela intre concepte a usurat foarte mult implementarea programului. Problema fiind, dupa parerea noastra, un caz particular al problemei Producator-Consumator, cu constrangerea prioritatii diferite a thread-urilor.

Consideratii pentru implementare

- problema de tipul RTOS environment
- timpul de 10 minute inseamna o secunda in program
- asemanatoare problemei tipice Producer-Cosumer (producer-pacienti/consumer-cabinet//buffer-queue)
- implementam sincronizarea cu semafoare, mutexuri si bariere
- toate taskurile se executa periodic
- 5 task-uri: G, M, R, addQueue, monitorizare; G are prioritatea cea mai mare, iar R cea mai redusa
- 5 thread-uri:
  - P - pacienti : addQueue
  - C - cabient : G, M, R
  - M - monitorizare
- resursa comuna: Q - queue (coada de asteptare)

3 Concepte de programare in timp real

Concepte teoretice

Conceptele prezentate la curs pe care le folosim:

- sincronizare
- excludere mutuală
- planificarea pacientilor pe condiţie de timp

Structuri folosite

Utilizam urmatoarele obiecte din teoria programarii in timp real si din categoria de sincronizare a thread-urilor:

- **2 mutexuri** : mutexCabinet (echivalent semafor binar) , mutexQueue
- **5 semafoare generalizate**:
  - sG :
    - >0 - in functie de cati G sunt in queue
    - 0 - niciun G in queue
  - sM :
    - >0 - in functie de cati M sunt in queue
    - 0 - niciun M in queue
  - sR :
    - >0 - in functie de cati R sunt in queue

    * 0 - niciun R in queue
  - occupiedSeats - numara cate locuri ocupate sunt in sala de asteptare
  - emptySeats - numara cate locuri libere sunt ramase in sala de asteptare

- **3 semafoare binare**:
  - Monitor - semaforul activeaza monitorizarea pacientilor dupa eliberarea cabinetului
  - barrierM - blocheaza sau permite accesul pacientilor M
  - barrierR - blocheaza sau permite accesul pacientilor R

- **variabile globale**:
  - isG :
    * 1 - sunt G sunt in sala
    * 0 - niciun G in sala
  - isM :
    * 1 - sunt M sunt in sala
    * 0 - niciun M in sala
  - isBarrier - bariera aceasta se activeaza cand una din barierele M sau R sunt active

- **valori initiale**:
  - sG, sM, sR, occupiedSeats, isG, isM, barrierM, barrierR, isBarrier = 0;
  - emptySeats = 10;
  - Monitor = 1;

4 Implementarea soluţiei

Explicare pas cu pas a functiilor programului

1. Task addQueue - Adaug pacienti G, M, R in coada de asteptare
   - creez un numar random de prioritate: 1,2 sau 3
   - astept loc disponibil in coada de asteptare
   - blochez accesul la coada de asteptare pentru a adauga un nou pacient
   - astept un timp random pt a veni un nou pacient
2. Task G - Preluare in Cabinet pacient cu prioritate medicala grava:
   - este G in cabinet =¿ Nu!
   - verific daca sunt pacienti G in coada de asteptare si decrementez dc exista
   - este G in cabinet =¿ Da!
   - verific daca cabinetul este liber si se ocupa cu pacientul G
   - ——Shared resource——
   - consultare pacient G
   - cabinetul devine liber pentru orice alt pacient
   - incrementez numarul de scaune libere din coada de asteptare
   - cabinetul devine liber pentru orice alt pacient =¿ monitorizare (aleg alt pacient)
3. Task M - Preluare in Cabinet pacient cu prioritate medicala medie:
   - este M in cabinet =¿ Nu!
   - verific daca sunt M in coada de asteptare si decrementez dc exista
   - verific daca nu sunt G in coada de asteptare prin barrierM
   - este M in cabinet =¿ Da!
   - verific daca cabinetul este liber si se ocupa cu pacientul M
   - ——Shared resource——
   - consultare pacient M
   - cabinetul devine liber pentru orice alt pacient
   - incrementez numarul de scaune libere din coada de asteptare
   - cabinetul devine liber pentru orice alt pacient =¿ monitorizare (aleg alt pacient)
4. Task R - Preluare in Cabinet pacient cu prioritate medicala redusa:
   - verific daca sunt R in coada de asteptare si decrementez dc exista
   - pacientul R asteapta 5 minute din politete pentru cei gravi
   - verific daca nu sunt G sau M in coada de asteptare (barrierR)
   - verific daca cabinetul este liber si se ocupa cu pacientul R
   - ——Shared resource——
   - consultare pacient R
   - cabinetul devine liber pentru orice alt pacient
   - incrementez numarul de scaune libere din coada de asteptare
   - cabinetul devine liber pentru orice alt pacient =¿ monitorizare (aleg alt pacient)
5. Task Monitorizare - Decide intr-un timp de 3 minute(0.3 secunde) ce pacient va intra in cabinet
   - se declanseaza cand cabinetul devine gol

- daca dupa 0.15 secunde nu a intrat nici-un G in salon, adica isG este 0 atunci M poate sa intre
- daca dupa inca 0.15 sec nu intra niciun M, atunci intra R
- numara cati pacienti au intrat in cabinet
- conditia de iesire: au intrat 10 pacienti

Organigrama

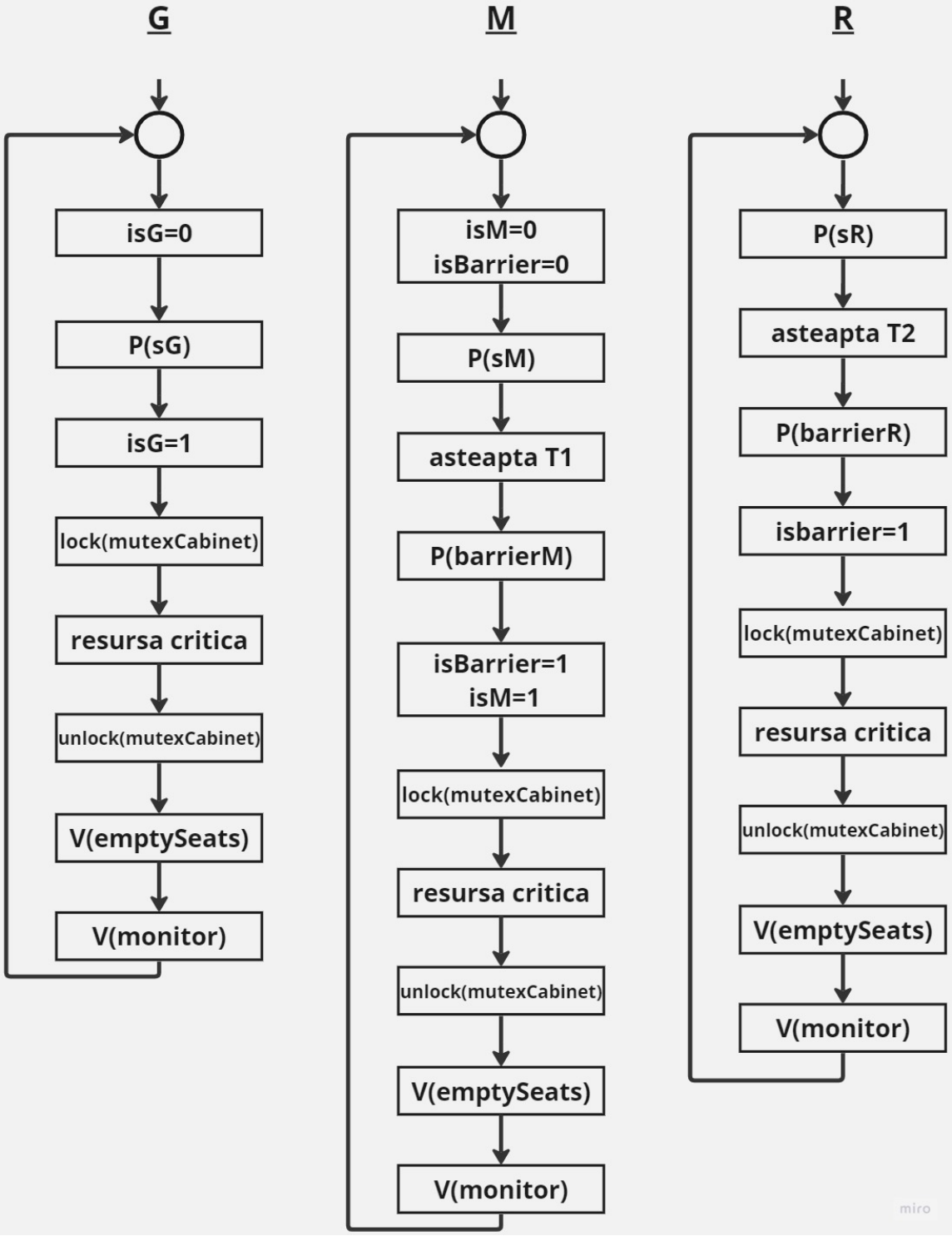

Figure 1: Soluţie implementare - organigrama taskuri

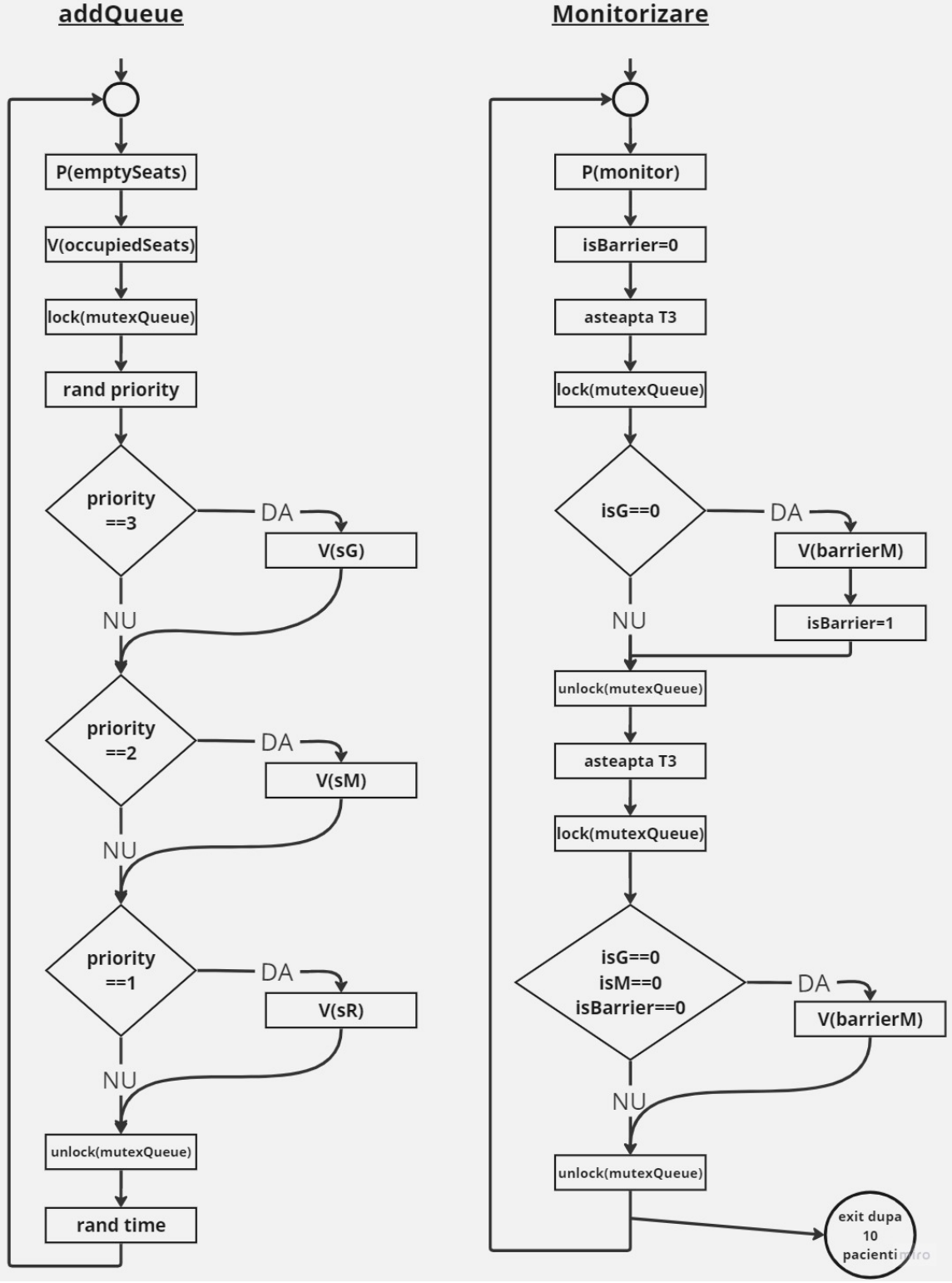

Figure 2: Soluţie implementare - organigrama taskuri

5 Codul programului

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

#define THREAD_NUM 5

//semafoare si mutex-uri:
sem_t sG, sM, sR;
sem_t emptySeats, occupiedSeats, monitor;
sem_t barrierM, barrierR;
pthread_mutex_t mutexQueue, mutexCabinet;

int isG = 0;
int isM = 0; //val conditie 0: exista; 1: nu exista pacient
int isBarrier = 0;
int count = 0;


void* addQueue (void* addQueue){
for (int i = 0; i<=15; i++){
//random number 1,2,3:
int priority = rand() % 3 + 1;

sem_wait(&emptySeats);
sem_post(&occupiedSeats);

pthread_mutex_lock(&mutexQueue);
//add pacient Grav: G
if (priority == 3){
sem_post(&sG);
// Pacienti
        printf("<---------------------------Pacient 'G' a intrat in sala de asteptare\n");
}

//add pacient Mediu: M
if (priority == 2){
sem_post(&sM);
// Pacienti
        printf("<---------------------------Pacient 'M' a intrat in sala de asteptare\n");

}

//add pacient Redus: R
if (priority == 1){
sem_post(&sR);
// Pacienti
        printf("<---------------------------Pacient 'R' a intrat in sala de asteptare\n");

}
pthread_mutex_unlock(&mutexQueue);
```

```
float timp_adaugare = rand() /(float)RAND_MAX * 6 + 0.5;
sleep(timp_adaugare);
}
}

void* G (void* args){
while (1){
isG = 0; //niciun G in coada

sem_wait(&sG); //exista G, intra in cabinet

isG = 1; //G in cabinet

sleep(0.1); //timpul de parcurs pana in cabinet

pthread_mutex_lock(&mutexCabinet);
//---shared resource---
printf("\nnr.%d Pacientul G a intrat in cabinet!\n", count);
float timp_consultare = rand() /(float)RAND_MAX * 4.5 + 4.5;
sleep ( timp_consultare);
printf("Pacientul 'G' a iesit din cabinet dupa %.2f min\n", timp_consultare*10);
pthread_mutex_unlock(&mutexCabinet);

sleep(0.1); //timpul de realizare cabinet gol

sem_post(&emptySeats);
sem_post(&monitor);
}
}

void* M (void* args){
while (1){
//sem_wait(&barrierM); //Nu este G, M poate intra
isM = 0; //niciun M in coada
isBarrier = 0; //bariera este deschisa pentru R pt ca M nu a intrat in cabinet

sem_wait(&sM); //Exista M in coada, asteapta sa intre in cabinet
sem_wait(&barrierM); //Nu este G, M poate intra
isM = 1; //M in cabinet
isBarrier = 1; //bariera inchisa pentru R
//printf("test->bariera inchisa pentru R!\n");

sleep(0.1); //timpul de parcurs pana in cabinet

pthread_mutex_lock(&mutexCabinet);
//---shared resource---
printf("\nnr.%d Pacientul M a intrat in cabinet!\n", count);
float timp_consultare = rand() /(float)RAND_MAX * 3 + 3;
sleep ( timp_consultare);
printf("Pacientul 'M' a iesit din cabinet dupa %.2f min\n", timp_consultare*10);
pthread_mutex_unlock(&mutexCabinet);

sleep(0.1); //timpul de realizare cabinet gol
sem_post(&emptySeats);
```

```
sem_post(&monitor);

}
}

void* R (void* args){
while (1){
sem_wait(&sR); //Exista R in coada, asteapta sa intre in cabinet
sleep(0.5); //timpul de asteptare raspuns monitorizare pacienti
sem_wait(&barrierR); //Nu este nici M, nici G, R poate intra

sleep(0.1); //timpul de parcurs pana in cabinet

pthread_mutex_lock(&mutexCabinet);
//---shared resource---
printf("\nnr.%d Pacientul R a intrat in cabinet!\n", count);
float timp_consultare = rand() /(float)RAND_MAX * 2 + 1;
sleep (timp_consultare);
printf("Pacientul 'R' a iesit din cabinet dupa %.2f min\n", timp_consultare*10);
pthread_mutex_unlock(&mutexCabinet);

sleep(0.1); //timpul de realizare cabinet gol

sem_post(&emptySeats);
sem_post(&monitor);

}
}

void* monitorizare (void* args){
while (1){
sem_wait(&monitor); //usa cabinetului s-a deschis, se cauta pacientul
isBarrier = 0; //M nu a intrat, R poate intra
sleep(0.15); //asteapta sa se gaseasca G in coada

pthread_mutex_lock(&mutexQueue);
if (isG == 0){ //nu e G si niciun G in asteptare
sem_post(&barrierM); //M poate intra, G nu este
//printf("test->bariera M ridicata!\n");
}
pthread_mutex_unlock(&mutexQueue);

sleep (0.15); //asteapta sa se gaseasca M in coada

pthread_mutex_lock(&mutexQueue);
if (isG == 0 && isM == 0 && isBarrier == 0){ //nu e G, nici M si nici M in asteptare
sem_post(&barrierR); //R poate intra, G si M nu sunt
//printf("test->bariera R ridicata!\n");
}
pthread_mutex_unlock(&mutexQueue);
//exit conditions:
count++;
if (count == 10){
sleep(6);
exit(0);
```

```
}
}
}


void* (*functie [])(void*) = {addQueue, G, M, R, monitorizare};

int main(int argc, char* argv[]){
int i;
srand(time(NULL));
pthread_t th[THREAD_NUM];

    sem_init(&sG, 0, 0);
    sem_init(&sM, 0, 0);
    sem_init(&sR, 0, 0);
sem_init(&occupiedSeats, 0, 0);
    sem_init(&emptySeats, 0, 10);
sem_init(&monitor, 0, 1);
sem_init(&barrierM, 0, 0);
sem_init(&barrierR, 0, 0);

    pthread_mutex_init(&mutexQueue, NULL);
pthread_mutex_init(&mutexCabinet, NULL);

for (i = 0; i < THREAD_NUM; i++){
        if (pthread_create(&th[i], NULL,  (void*)(*(functie+i)), NULL) != 0){
            perror("Failed to create thread");
        }
    }

    for (i = 0; i < THREAD_NUM; i++){
        if (pthread_join(th[i], NULL) != 0){
            perror("Failed to join thread");
        }
    }

    sem_destroy(&sG);
    sem_destroy(&sM);
    sem_destroy(&sR);
sem_destroy(&occupiedSeats);
    sem_destroy(&emptySeats);
    sem_destroy(&monitor);
sem_destroy(&barrierM);
sem_destroy(&barrierR);
    pthread_mutex_destroy(&mutexQueue);
pthread_mutex_destroy(&mutexCabinet);
    pthread_exit(NULL);
}
```

6 Testarea aplicaţiei

Testari reusite ale codului.

Date1:

```
<------------------Pacient 'G' a intrat in sala de asteptare
noG:0, noM:0

Pacientul G a intrat in cabinet!
<------------------Pacient 'G' a intrat in sala de asteptare
Pacientul 'G' a iesit din cabinet dupa 6.70
noG:0, noM:0

Pacientul G a intrat in cabinet!
<------------------Pacient 'R' a intrat in sala de asteptare
<------------------Pacient 'R' a intrat in sala de asteptare
Pacientul 'G' a iesit din cabinet dupa 10.67
noG:0, noM:0

Pacientul R a intrat in cabinet!
<------------------Pacient 'R' a intrat in sala de asteptare
Pacientul 'R' a iesit din cabinet dupa 2.84
noG:0, noM:0
<------------------Pacient 'G' a intrat in sala de asteptare

Pacientul G a intrat in cabinet!
<------------------Pacient 'R' a intrat in sala de asteptare
<------------------Pacient 'R' a intrat in sala de asteptare
Pacientul 'G' a iesit din cabinet dupa 11.12
noG:0, noM:0
<------------------Pacient 'R' a intrat in sala de asteptare

Pacientul R a intrat in cabinet!
<------------------Pacient 'R' a intrat in sala de asteptare
Pacientul 'R' a iesit din cabinet dupa 2.33
noG:0, noM:0

Pacientul R a intrat in cabinet!
Pacientul 'R' a iesit din cabinet dupa 1.32
noG:0, noM:0

Pacientul R a intrat in cabinet!
Pacientul 'R' a iesit din cabinet dupa 1.53
noG:0, noM:0

Pacientul R a intrat in cabinet!
Pacientul 'R' a iesit din cabinet dupa 2.63
noG:0, noM:0

Pacientul R a intrat in cabinet!
Pacientul 'R' a iesit din cabinet dupa 1.81
noG:0, noM:0

Pacientul R a intrat in cabinet!
```

```
Pacientul 'R' a iesit din cabinet dupa 2.15
noG:0, noM:0
^Z
```

Date2:

```
<---------------------------Pacient 'G' a intrat in sala de asteptare
<---------------------------Pacient 'R' a intrat in sala de asteptare

Pacientul G a intrat in cabinet!
noG:1, noM:0
<---------------------------Pacient 'G' a intrat in sala de asteptare
Pacientul 'G' a iesit din cabinet dupa 31.60 min
noG:0, noM:0

Pacientul G a intrat in cabinet!
<---------------------------Pacient 'R' a intrat in sala de asteptare
<---------------------------Pacient 'M' a intrat in sala de asteptare
<---------------------------Pacient 'G' a intrat in sala de asteptare
<---------------------------Pacient 'M' a intrat in sala de asteptare
Pacientul 'G' a iesit din cabinet dupa 69.08 min
<---------------------------Pacient 'R' a intrat in sala de asteptare
noG:0, noM:0

Pacientul G a intrat in cabinet!
<---------------------------Pacient 'G' a intrat in sala de asteptare
<---------------------------Pacient 'R' a intrat in sala de asteptare
Pacientul 'G' a iesit din cabinet dupa 41.82 min
noG:0, noM:0

Pacientul G a intrat in cabinet!
Pacientul 'G' a iesit din cabinet dupa 36.57 min
noG:0, noM:0

Pacientul M a intrat in cabinet!
Pacientul 'M' a iesit din cabinet dupa 42.09 min
noG:0, noM:0

Pacientul M a intrat in cabinet!
Pacientul 'M' a iesit din cabinet dupa 49.70 min

Pacientul R a intrat in cabinet!
noG:0, noM:0
Pacientul 'R' a iesit din cabinet dupa 25.42 min
noG:0, noM:0

Pacientul R a intrat in cabinet!
Pacientul 'R' a iesit din cabinet dupa 15.23 min
noG:0, noM:0

Pacientul R a intrat in cabinet!
Pacientul 'R' a iesit din cabinet dupa 19.60 min
noG:0, noM:0

Pacientul R a intrat in cabinet!
Pacientul 'R' a iesit din cabinet dupa 26.98 min
```

```
noG:0, noM:0
```

Date3:

```
<---------------------------Pacient 'G' a intrat in sala de asteptare

Pacientul G a intrat in cabinet!
<---------------------------Pacient 'R' a intrat in sala de asteptare
<---------------------------Pacient 'G' a intrat in sala de asteptare
<---------------------------Pacient 'G' a intrat in sala de asteptare
Pacientul 'G' a iesit din cabinet dupa 69.85 min

Pacientul G a intrat in cabinet!
<---------------------------Pacient 'R' a intrat in sala de asteptare
Pacientul 'G' a iesit din cabinet dupa 60.53 min

Pacientul G a intrat in cabinet!
<---------------------------Pacient 'M' a intrat in sala de asteptare
<---------------------------Pacient 'M' a intrat in sala de asteptare
<---------------------------Pacient 'M' a intrat in sala de asteptare
<---------------------------Pacient 'R' a intrat in sala de asteptare
<---------------------------Pacient 'G' a intrat in sala de asteptare
Pacientul 'G' a iesit din cabinet dupa 115.06 min

Pacientul G a intrat in cabinet!
Pacientul 'G' a iesit din cabinet dupa 87.61 min

Pacientul M a intrat in cabinet!
Pacientul 'M' a iesit din cabinet dupa 46.22 min

Pacientul M a intrat in cabinet!
Pacientul 'M' a iesit din cabinet dupa 56.08 min

Pacientul M a intrat in cabinet!
Pacientul 'M' a iesit din cabinet dupa 39.94 min

Pacientul R a intrat in cabinet!
Pacientul 'R' a iesit din cabinet dupa 10.63 min

Pacientul R a intrat in cabinet!
Pacientul 'R' a iesit din cabinet dupa 12.00 min

Pacientul R a intrat in cabinet!
Pacientul 'R' a iesit din cabinet dupa 15.94 min
```

Erori identificate si functionalitate malitioasa

- Deadlocks : nu sunt;
- Race conditions : sunt prezente in cadrul resursei comune: adica a cozii de asteptare: G, M, R ; de aici rezulta in unele teste comportament nedefinit;
- Erori de sincronizare : provin de la functia sleep, de la race conditions, de la incetinirea CPU, sau problema de arhitectura. In orice caz, apar foarte rar, si incurca randul solutiei bune, cu maxim 2 sau 3 valori din 10, cate are o executare.
- Functionalitate malitioasa : din mai mult de 50 de teste facut in 5-10 teste am descoperit o comportare nedorita, de tipul erorilor de sincronizare discutate mai sus.

In continuare sunt prezentate aceste tipuri de erori de sincronizare, care apar rar, impreuna cu printarea valorilor variabilelor bariera pentru debugging.

Eroare1:

Solutia asteptata: M,G,R,R,R,R,G,G,M,R . Solutia calculata de program: M,G,R,R,R,R,G,M,G,R

```
<--------------------------Pacient 'M' a intrat in sala de asteptare
test->bariera M ridicata!
test->bariera inchisa pentru R!

Pacientul M a intrat in cabinet!
<--------------------------Pacient 'G' a intrat in sala de asteptare
Pacientul 'M' a iesit din cabinet dupa 43.40 min

Pacientul G a intrat in cabinet!
<--------------------------Pacient 'R' a intrat in sala de asteptare
Pacientul 'G' a iesit din cabinet dupa 53.10 min
test->bariera M ridicata!
test->bariera R ridicata!

Pacientul R a intrat in cabinet!
<--------------------------Pacient 'R' a intrat in sala de asteptare
Pacientul 'R' a iesit din cabinet dupa 29.48 min
test->bariera M ridicata!
test->bariera R ridicata!

Pacientul R a intrat in cabinet!
<--------------------------Pacient 'R' a intrat in sala de asteptare
Pacientul 'R' a iesit din cabinet dupa 19.28 min
test->bariera M ridicata!
test->bariera R ridicata!

Pacientul R a intrat in cabinet!
Pacientul 'R' a iesit din cabinet dupa 13.57 min
test->bariera M ridicata!
test->bariera R ridicata!
<--------------------------Pacient 'R' a intrat in sala de asteptare

Pacientul R a intrat in cabinet!
<--------------------------Pacient 'G' a intrat in sala de asteptare
Pacientul 'R' a iesit din cabinet dupa 18.92 min
```

```
Pacientul G a intrat in cabinet!
<--------------------------Pacient 'R' a intrat in sala de asteptare
<--------------------------Pacient 'G' a intrat in sala de asteptare
<--------------------------Pacient 'M' a intrat in sala de asteptare
test->bariera inchisa pentru R!
Pacientul 'G' a iesit din cabinet dupa 73.14 min

Pacientul M a intrat in cabinet!
Pacientul 'M' a iesit din cabinet dupa 47.62 min

Pacientul G a intrat in cabinet!
Pacientul 'G' a iesit din cabinet dupa 69.13 min
test->bariera M ridicata!
test->bariera R ridicata!

Pacientul R a intrat in cabinet!
Pacientul 'R' a iesit din cabinet dupa 27.03 min
test->bariera M ridicata!
test->bariera R ridicata!
```

Eroare1 rezolvata:

Pe un caz asemanator, cu modificarea codului programului, eroarea aseamanatoare cazului anterior, nu a mai aparut.

```
<--------------------------Pacient 'R' a intrat in sala de asteptare
test->bariera M ridicata!
test->bariera R ridicata!

Pacientul R a intrat in cabinet!
Pacientul 'R' a iesit din cabinet dupa 25.54 min
test->bariera M ridicata!
test->bariera R ridicata!
<--------------------------Pacient 'R' a intrat in sala de asteptare

Pacientul R a intrat in cabinet!
Pacientul 'R' a iesit din cabinet dupa 10.33 min
test->bariera M ridicata!
test->bariera R ridicata!
<--------------------------Pacient 'G' a intrat in sala de asteptare

Pacientul G a intrat in cabinet!
<--------------------------Pacient 'M' a intrat in sala de asteptare
test->bariera inchisa pentru R!
<--------------------------Pacient 'R' a intrat in sala de asteptare
Pacientul 'G' a iesit din cabinet dupa 86.20 min

Pacientul M a intrat in cabinet!
test->bariera M ridicata!
<--------------------------Pacient 'R' a intrat in sala de asteptare
Pacientul 'M' a iesit din cabinet dupa 43.25 min

Pacientul R a intrat in cabinet!
test->bariera M ridicata!
test->bariera R ridicata!
```

```
Pacientul 'R' a iesit din cabinet dupa 14.80 min
test->bariera M ridicata!

Pacientul R a intrat in cabinet!
test->bariera R ridicata!
<--------------------------Pacient 'M' a intrat in sala de asteptare
test->bariera inchisa pentru R!
<--------------------------Pacient 'M' a intrat in sala de asteptare
Pacientul 'R' a iesit din cabinet dupa 15.43 min

Pacientul M a intrat in cabinet!
<--------------------------Pacient 'G' a intrat in sala de asteptare
test->bariera M ridicata!
Pacientul 'M' a iesit din cabinet dupa 31.47 min

Pacientul G a intrat in cabinet!
test->bariera inchisa pentru R!
<--------------------------Pacient 'M' a intrat in sala de asteptare
Pacientul 'G' a iesit din cabinet dupa 69.89 min

Pacientul M a intrat in cabinet!
test->bariera M ridicata!
Pacientul 'M' a iesit din cabinet dupa 52.45 min
test->bariera inchisa pentru R!
test->bariera M ridicata!

Pacientul M a intrat in cabinet!
Pacientul 'M' a iesit din cabinet dupa 46.68 min
test->bariera M ridicata!
test->bariera R ridicata!
```

Eroare de sincronizare asemanatoare primei erorii, nerezolvate. Solutia buna era R,M,M,G,G,G,G,R,R,R, solutia gasita de program: R,M,M,G,R,G,G,G,R,R.

Eroare2:

```
<--------------------------Pacient 'R' a intrat in sala de asteptare
test->bariera M ridicata!
test->bariera R ridicata!

Pacientul R a intrat in cabinet!
Pacientul 'R' a iesit din cabinet dupa 27.27 min
test->bariera M ridicata!
test->bariera R ridicata!
<--------------------------Pacient 'M' a intrat in sala de asteptare
test->bariera inchisa pentru R!

Pacientul M a intrat in cabinet!
<--------------------------Pacient 'M' a intrat in sala de asteptare
Pacientul 'M' a iesit din cabinet dupa 56.52 min
test->bariera inchisa pentru R!
test->bariera M ridicata!

Pacientul M a intrat in cabinet!
<--------------------------Pacient 'G' a intrat in sala de asteptare
Pacientul 'M' a iesit din cabinet dupa 40.40 min
```

```
<---------------------------Pacient 'G' a intrat in sala de asteptare

Pacientul G a intrat in cabinet!
<---------------------------Pacient 'R' a intrat in sala de asteptare
<---------------------------Pacient 'G' a intrat in sala de asteptare
Pacientul 'G' a iesit din cabinet dupa 79.81 min

Pacientul R a intrat in cabinet!
Pacientul 'R' a iesit din cabinet dupa 28.87 min

Pacientul G a intrat in cabinet!
<---------------------------Pacient 'G' a intrat in sala de asteptare
<---------------------------Pacient 'R' a intrat in sala de asteptare
Pacientul 'G' a iesit din cabinet dupa 65.17 min

Pacientul G a intrat in cabinet!
<---------------------------Pacient 'R' a intrat in sala de asteptare
Pacientul 'G' a iesit din cabinet dupa 64.28 min

Pacientul G a intrat in cabinet!
Pacientul 'G' a iesit din cabinet dupa 45.96 min
test->bariera M ridicata!
test->bariera R ridicata!

Pacientul R a intrat in cabinet!
Pacientul 'R' a iesit din cabinet dupa 23.60 min
test->bariera M ridicata!
test->bariera R ridicata!

Pacientul R a intrat in cabinet!
Pacientul 'R' a iesit din cabinet dupa 29.67 min
test->bariera M ridicata!
test->bariera R ridicata!
```

Validarea solutiei propuse

Verificare solutie 1:

Va prezentam doua solutii corecte gasite de program, pentru valori de intrare random si "mai grele" decat majoritatea valorilor de intrare random.

```
<--------------------------Pacient 'M' a intrat in sala de asteptare
<--------------------------Pacient 'M' a intrat in sala de asteptare
<--------------------------Pacient 'R' a intrat in sala de asteptare
test->bariera M ridicata!
test->bariera inchisa pentru R!

Pacientul M a intrat in cabinet!
<--------------------------Pacient 'M' a intrat in sala de asteptare
Pacientul 'M' a iesit din cabinet dupa 43.25 min
test->bariera M ridicata!
test->bariera inchisa pentru R!

Pacientul M a intrat in cabinet!
<--------------------------Pacient 'M' a intrat in sala de asteptare
Pacientul 'M' a iesit din cabinet dupa 41.17 min
test->bariera M ridicata!
test->bariera inchisa pentru R!

Pacientul M a intrat in cabinet!
<--------------------------Pacient 'G' a intrat in sala de asteptare
Pacientul 'M' a iesit din cabinet dupa 48.07 min

Pacientul G a intrat in cabinet!
<--------------------------Pacient 'G' a intrat in sala de asteptare
<--------------------------Pacient 'R' a intrat in sala de asteptare
<--------------------------Pacient 'G' a intrat in sala de asteptare
<--------------------------Pacient 'M' a intrat in sala de asteptare
Pacientul 'G' a iesit din cabinet dupa 85.78 min

Pacientul G a intrat in cabinet!
Pacientul 'G' a iesit din cabinet dupa 62.53 min

Pacientul G a intrat in cabinet!
Pacientul 'G' a iesit din cabinet dupa 82.42 min
test->bariera M ridicata!
test->bariera inchisa pentru R!

Pacientul M a intrat in cabinet!
Pacientul 'M' a iesit din cabinet dupa 41.76 min
test->bariera M ridicata!
test->bariera inchisa pentru R!

Pacientul M a intrat in cabinet!
Pacientul 'M' a iesit din cabinet dupa 30.98 min
test->bariera M ridicata!
test->bariera R ridicata!

Pacientul R a intrat in cabinet!
Pacientul 'R' a iesit din cabinet dupa 12.60 min
```

```
test->bariera M ridicata!
test->bariera R ridicata!

Pacientul R a intrat in cabinet!
Pacientul 'R' a iesit din cabinet dupa 26.03 min
test->bariera M ridicata!
test->bariera R ridicata!
```

Verificare solutie 2:

```
<--------------------------Pacient 'M' a intrat in sala de asteptare
<--------------------------Pacient 'G' a intrat in sala de asteptare
<--------------------------Pacient 'G' a intrat in sala de asteptare

Pacientul G a intrat in cabinet!
<--------------------------Pacient 'M' a intrat in sala de asteptare
<--------------------------Pacient 'M' a intrat in sala de asteptare
<--------------------------Pacient 'M' a intrat in sala de asteptare
Pacientul 'G' a iesit din cabinet dupa 85.23 min
<--------------------------Pacient 'M' a intrat in sala de asteptare

Pacientul G a intrat in cabinet!
<--------------------------Pacient 'R' a intrat in sala de asteptare
Pacientul 'G' a iesit din cabinet dupa 56.89 min
<--------------------------Pacient 'R' a intrat in sala de asteptare
test->bariera M ridicata!
test->bariera inchisa pentru R!

Pacientul M a intrat in cabinet!
<--------------------------Pacient 'G' a intrat in sala de asteptare
Pacientul 'M' a iesit din cabinet dupa 53.01 min

Pacientul G a intrat in cabinet!
Pacientul 'G' a iesit din cabinet dupa 71.02 min
test->bariera M ridicata!
test->bariera inchisa pentru R!

Pacientul M a intrat in cabinet!
Pacientul 'M' a iesit din cabinet dupa 32.26 min
test->bariera M ridicata!
test->bariera inchisa pentru R!

Pacientul M a intrat in cabinet!
Pacientul 'M' a iesit din cabinet dupa 37.25 min
test->bariera M ridicata!
test->bariera inchisa pentru R!

Pacientul M a intrat in cabinet!
Pacientul 'M' a iesit din cabinet dupa 30.89 min
test->bariera M ridicata!
test->bariera inchisa pentru R!

Pacientul M a intrat in cabinet!
Pacientul 'M' a iesit din cabinet dupa 51.42 min
test->bariera M ridicata!
test->bariera R ridicata!
```

```
Pacientul R a intrat in cabinet!
Pacientul 'R' a iesit din cabinet dupa 14.21 min
test->bariera M ridicata!
test->bariera R ridicata!

Pacientul R a intrat in cabinet!
Pacientul 'R' a iesit din cabinet dupa 23.30 min
test->bariera M ridicata!
test->bariera R ridicata!
```

Metode verificare solutii

Pentru analiza solutiei, trebuie tinut cont de momentul de timp cand intra pacientii in coada de asteptare si de momentul de timp cand cabinetul se elibereaza. Se poate urmari la executarea programului sau cu ajutorul unui timeline chart de tipul celui de mai jos sau de tipul Gantt chart.

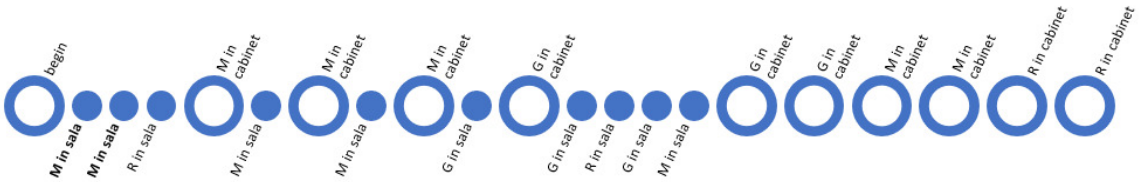

Figure 3: Timeline chart Verificare solutie 1

Solutie asteptata: MMMGGGMMRR
Solutia verifica problema.

7 Concluzii

Pentru Problema Prioritatii Pacientilor am gasit o solutie de sincronizare, pe care am denumit-o "Politetea pacientilor". Solutia este de tipul programarii in timp real, si utilizeaza concepte de RTOS, sincronizare si excludere mutuala. Noi am considerat problema un caz particular a problemei arhetip in sincronizare: "Producer-Consumer", prin existenta prioritatii diferite a thread-urilor, care a reprezentat si cea mai mare dificultate a problemei.

Rezultatul programului implementat este de cele mai multe ori corect, cu greseli ce apar rar si sunt incluse in categoria eroriilor de sincronizare. Fiind singurul tip de erori intalnite la testare, pe un numar de aproximativ 50 de executari, acestea pot aparea datorita greselilor de sincronizare a thread-urilor, incetinirii CPU, race conditions sau arhitecturii programului.

Calculand eroarea rezultatului, raportata la numarul de valori, din cele 50 de teste: numarul de valori gresite din 500 de valori, este cuprins intre 20-30. Deci probabilitatea ca o valoare din sirul de valori dintr-o executare sa fie eronata este de 4%, iar numarul de teste gresite din 50, este 10%.

In concluzie, 90% din teste rezolva corect problema, in timp ce 96% din valorile returnate coincid cu valoriile solutiei problemei.

8 Bibliografie

https://en.wikipedia.org/wiki/Producer%E2%80%93consumer_problem

https://code-vault.net/lesson/18ec1942c2da46840693efe9b520f873
https://www.geeksforgeeks.org/use-posix-semaphores-c/
https://linuxhint.com/https-linuxhint-com-posix-semaphores-with-c-programming/
https://greenteapress.com/wp/semaphores/