Wiki »

Exercise5 Thread Synchronization II - Parking lot control system

Indledning

I denne opgave vil vi implementere et Parking Lot Control System (PLCS) som styrer biler der vil ind og ud på p-pladsen. Først vil vi designe vores løsning for den første del af øvelsen ved at skrive pseudokode, dernæst implementere pseudokoden og tjekke at programmet kører som forventet. Derefter udvider vi programmet til at kunne håndtere flere biler og til sidst implementerer vi således at p-pladsen har et maks antal biler som kan være derinde, så bilerne ikke kan komme ind hvis p-pladsen er fyldt.

De følgende opgaver løses ved brug af mutexes og conditionals, og ikke semaphores

Exercise1 - Implement Park-a-lot 2000 - Design(%20) ¶

Herunder ses et udkast til hvordan opgave 1 kan løses i form af pseudokode:

```
carWaitingToGetOut = false;
carWaitingToGetIn = false;
numberOfCars = 0;

mutex mutex;
cond condentry, condexit;
entryopen = false;
exitopen = false;

entry()
{
  while(1)
  {
    Lock mutex;
    while(!carWaitingToGetIn)
    {
      condwait(condentry,mutex);
    }
    entryopen = true;
    condsignal(condentry);

    print("Car has been parked");

    carWaitingToGetIn = false;
    carWaitingToGetOut = true;
    numberOfCars = 1;
    Unlock mutex;
  }
  return null;
}

exit()
```

```
{
  while(1)
  {
      Lock mutex;
      while(!carWaitingToGetOut)
      {
        condwait(condexit,mutex);
      }

      print("Car has exited parkinglot");
      carWaitingToGetOut = false;
      carWaitingToGetIn = true;
      numberOfCars = 0;

      Unlock mutex;
  }
  return null;
}

car()
{
  while(1)
  {
    if(numberOfCars == 0)
    {
      Lock mutex;
      carWaitingToGetIn = true;
      condsignal(condentry);

      Unlock mutex;
      waitForSomeTime(); //vent inde på p-plads
    }
    else if(numberOfCars == 1)
    {
      Lock mutex;
      carWaitingToGetOut = true;
      condsignal(condexit);

      Unlock mutex;
      waitForSomeTime(); //vent ude for p-plads
    }
  }
  return null;
}

main()
{
  initier conditionals;
  initier mutex;

  create threads;

  join threads;

  return 0;

}
```

Dette bliver vores grundlag for de følgende opgaver

Exercise2 - Implement Park-a-lot 2000 - Implementation(%40)

Exercise2.1 - First step

Koden for park-a-lot-2000 er lavet ud fra den pseudokode, som blev designet i exercise 1. Den består altså af 3 metoder: en car, en entry og en exit. car metoden signalerer de to andrer vha. pthread_cond_signal() når den skal enten ind eller ud og den bliver så lukket igennem porten. når en metode venter den på et signal står den i en whileløkke hvori der er en pthread_cond_wait(), som venter på at blive signaleret fra en af de andre metoder. koden for car metoden kan ses herunder, resten af source koden kan ses i vores repository.

```
void *car(void*){
  for(;;){
    if(car1_in_garage==0)
    {
      pthread_mutex_lock(&mut);
      Awaiting_Entry = true;
      pthread_cond_signal(&conentry);
      pthread_mutex_unlock(&mut);
      sleep(5);
    } else if(car1_in_garage==1)
    {
      pthread_mutex_lock(&mut);
      Awaiting_Exit = true;
      pthread_cond_signal(&conexit);
      pthread_mutex_unlock(&mut);
      sleep(5);
    }
  }
  return nullptr;
}
```

Da det kun skal være muligt for en bil at komme igennem i denne opgave er der lavet en global variabel "car1_in_garage", som holder styr på om bilen er i garagen og får metoderne til at agere efter det.

billede af terminalvinduet ved kørsel af programmet:

```
stud@stud-virtual-machine:~/Desktop/ISU/lektion5/exe2$ ./exe
Car has been parked
Car has exited the parkinglot
Car has been parked
Car has exited the parkinglot
^C
stud@stud-virtual-machine:~/Desktop/ISU/lektion5/exe2$
```

Exercise2.2 - The grandiose test

I denne opgave skulle koden fra den tidligere exercise udvides sådan, at der kan køre flere biler ind i garagen. Det har vi gjort ved at fjerne variablen "car1_in_garage" og fået entry og exit til, at signalere tilbage til bilen på en måde, så den kører ind, når den venter på at komme ind og kører ud når den venter på at køre ud. Herunder

ses den opdaterede kode for metoden car, resten af koden kan ses i repository:

```
17  void *car(void *arg){
18    srand(time(NULL));
19    while(1){
20      pthread_mutex_lock(&mut);
21      car_waiting_for_entry++;
22      cout << "Car " << pthread_self() << " is waiting to enter" << endl;
23      pthread_cond_signal(&conentry);
24      while(car_waiting_for_entry){
25        pthread_cond_wait(&carcond, &mut);
26      }
27      cout << "Car " << pthread_self() << " Has entered the parking lot" << endl;
28      pthread_mutex_unlock(&mut);
29
30      sleep(rand()%5+3);
31
32      car_waiting_for_exit++;
33
34      pthread_mutex_lock(&mut);
35      cout << "Car " << pthread_self() << " Is waiting to exit" << endl;
36      pthread_cond_signal(&conexit);
37      while(car_waiting_for_exit){
38        pthread_cond_wait(&carcond,&mut);
39      }
40      pthread_mutex_unlock(&mut);
41      cout << "Car " << pthread_self() << " Has exited the parking lot" << endl;
42
43    }
44    return nullptr;
45  }
```

billede af terminalvinduet ved kørsel af programmet:

```
stud@stud-virtual-machine:~/Desktop/ISU/lektion5/exe2.2$ ./exe
Car 140499675076352 is waiting to enter
Car 140499666683648 is waiting to enter
Car 140499666683648 Has entered the parking lot
Car 140499641505536 is waiting to enter
Car 140499675076352 Has entered the parking lot
Car 140499641505536 Has entered the parking lot
Car 140499649898240 is waiting to enter
Car 140499658290944 is waiting to enter
Car 140499658290944 Has entered the parking lot
Car 140499649898240 Has entered the parking lot
Car 140499675076352 Is waiting to exit
Car 140499641505536 Is waiting to exit
Car 140499641505536 Has exited the parking lot
Car 140499641505536 is waiting to enter
Car 140499666683648 Is waiting to exit
Car 140499666683648 Has exited the parking lot
Car 140499666683648 is waiting to enter
Car 140499649898240 Is waiting to exit
Car 140499649898240 Has exited the parking lot
Car 140499649898240 is waiting to enter
Car 140499675076352 Has exited the parking lot
Car 140499675076352 is waiting to enter
```

Det ses ud fra ovenstående kode at det eksekvere korrekt og efter ønsket måde. Dog ses det også at bilerne overhaler hinanden i køen, hvilket også er forklaret i de følgende afsnit.

Explain what pthread_cond_broadcast() does and argue as to why you needed or didnt need it.

Metoden pthread_cond_broadcast() ”unblocker” alle de blokerede tråde, som er låst i den samme ’conditional’ variabel. Vi har i vores implementering valgt ikke at bruge *broadcast()*, men har i stedet valgt at bruge *signal()* , som kan ”unblocke” enkelte eller flere tråde i stedet, hvilket var mere optimalt for den løsning vi gik med, så vi kan styre de forskellige tråde uafhængigt af hinanden.

The cars will seemingly not wait in line, but overtake each other on the way in or out. Why does this happen, and can you think of an approach that would fix it?

Dette sker fordi at der ikke holdes nogen orden for hvilken rækkefølge bilerne holder i kø. Derfor vælger scheduleren hvilke biler der får lov til hvad. Dette resulterer i at bilerne overtager hinandens ventepositioner, eller at de rettere sagt ”springer foran i køen”.
En mulig løsning på dette problem kunne være at lave en liste, i form af en heap eksempelvis, som prioriterer hvilke biler har ”førsteret” til at tilgå parkeringspladsen, som så senere ville unlocke for den tråd der var højest prioriteret.

Exercise3 - Extending PLCS, now with a limit on the number of cars (%40)

Der skal nu være en begrænsning på hvor mange biler der kan være i garagen og der er derfor kommet nogle nye globale variable: cars_in_garage, maxcars. Cars_in_garage inkrementerer hver gang der kører en bil ind og dekrementerer hver gang der kører en bil ud. Den kan ikke blive højere end maxcars. Ud over der er der i implementationen for car blevet tilføjet en while løkke, som venter på at cars_in_garage bliver mindre end maxcars og der derved kan køre en ny bil ind (kun hvis der er fyldt).
Det blev yderligere tydeligt i løbet af opgaven, at der var behov for endnu en mutex, så vi havde en der lukker biler ind og en som lukker biler ud. På den måde kan pkælderen stadig lukke biler ud selvom der står en bil og venter inde i et whileloop, som er inde i en låst entry mutex. Koden for implementeringen af car ses herunder,

resten af koden ligger i repository:

```
19  void *car(void *arg){
20    srand(time(NULL));
21    while(1){
22      pthread_mutex_lock(&mut);
23      car_waiting_for_entry++;
24      cout << "Car " << pthread_self() << " is waiting to enter" << endl;
25
26      pthread_cond_signal(&conentry);
27
28      while(car_waiting_for_entry){
29        pthread_cond_wait(&carcond, &mut);
30      }
31
32      while(cars_in_garage>=maxcars){
33      };
34      cars_in_garage++;
35      cout << cars_in_garage << endl;
36
37      cout << "Car " << pthread_self() << " Has entered the parking lot" << endl;
38      pthread_mutex_unlock(&mut);
39
40      sleep(rand()%5+3);
41
42      car_waiting_for_exit++;
43
44      pthread_mutex_lock(&mutexit);
45      cout << "Car " << pthread_self() << " Is waiting to exit" << endl;
46      pthread_cond_signal(&conexit);
47      cars_in_garage--;
48      pthread_mutex_unlock(&mutexit);
49      cout << "Car " << pthread_self() << " Has exited the parking lot" << endl;
50
51    }
52    return nullptr;
53  }
```

Extend your PLCS to handle this situation and verify that it actually does as you expect. Remember to test the scenario where a car leaves a full parking lot, enabling a waiting car to enter.

Bilen udskriver cars_in_garage til terminalen hver gang den er kørt ind, på den måde kan det verificeres, at der ikke er flere bilen i garagen end tilladt. Grænsen for antal biler er sat til 2. Billede af terminalvinduet ved kørsel af programmet:

```
stud@stud-virtual-machine:~/Desktop/ISU/lektion5/exe3$ ./exe
Car 139939038336768 is waiting to enter
Car 139938959980288 is waiting to enter
1
Car 139938959980288 Has entered the parking lot
Car 139938951587584 is waiting to enter
2
Car 139939038336768 Has entered the parking lot
Car 139938943194880 is waiting to enter
Car 139938959980288 Is waiting to exit
Car 139938959980288 Has exited the parking lot
2
Car 139938951587584 Has entered the parking lot
Car 139938934802176 is waiting to enter
Car 139939038336768 Is waiting to exit
Car 139939038336768 Has exited the parking lot
2
Car 139938934802176 Has entered the parking lot
```

Grunden til at der står der hele tiden er 2 biler i garagen er, at den ikke skriver ud når der kører en bil ud, men kun når der kører en ind. Derfor kører der en ud, så kører der en ind og så udskriver den.
Den skriver antallet af biler ud inden den skriver at bilen er kørt ind. Dette burde nok have været omvendt, men gør ikke noget for funktionaliteten.

- **Tilføjet af Ronnie Jakobsgaard for 3 dage siden**

  Vi har valgt de følgene fire fokuspunkter:

  - Forståelse for hele systemet ved hjælp af flowcharts/pseudo kode ✔
    - Gruppen viser at have en god forståelse for hvordan programmet skal virker og det viser de fint ved hjælp af pseudo kode.
  - Forståelse for condition variables og mutex ✔
    - I bruger conditions og mutex som man skal, locker og unlocker de rigtige steder.
    Det kan ses på konsol udskrifter samt kode at der er en tilstrækkelige forståelse for conditions og mutex til at løse opgaven.
  - Kontrol af tråde ✔
    Gruppen formår at kontrollere de forskellige tråde, og opnår herved den ønskede funktion af programmet.
    Dog kunne det være fedt hvis hver bil havede et id som ikke er 15 karakter langt.
  - Forsøg at lave en udvidelse af PLCS ✔
    Gruppen kommer fint i mål med udvidelsen af programmet. Programmet kan nu administrere et max antal biler og kan bekræftes af konsol udskriften

  ### feedback

  I har lavet en rigtig fin Wiki-side og kommer i mål med hele opgaven. Dog kunne det være fedt hvis hver bil har et id som ikke er mega langt så som 1, 2, og 3.
  Derudover kunne det være fint hvis koden bliver sat ind som tekst og ikke som billeder, det gør det nemmere at søge i evt.

  ### Konklusion

  Alt i alt en rigtig god opgave. Så derfor bliver opgaven noteret som OK her fra.

  ### Formelle krav

  Makefiles i repo ✔
  sourcecode i repo ✔

2.png (35,5 KB) Emil Christian Foged Klemmensen, 2019-03-19 13:52

3.png (94,1 KB) Emil Christian Foged Klemmensen, 2019-03-19 14:08

32.png (27,4 KB) Emil Christian Foged Klemmensen, 2019-03-19 14:11

222.png (32,6 KB) Emil Christian Foged Klemmensen, 2019-03-19 14:13

21.png (13,1 KB) Emil Christian Foged Klemmensen, 2019-03-19 14:14

22.png (86,9 KB) Emil Christian Foged Klemmensen, 2019-03-19 14:15