#include <stdio.h>

#include <sys/time.h>

#include <assert.h>

#include <pthread.h>

#include <assert.h>

#include <stdlib.h>

#include <unistd.h>

//double time;

double GetTime ()

{

struct timeval t;

int rc = gettimeofday (&t, NULL);

assert (rc == 0);

return (double) t.tv\_sec + (double) t.tv\_usec / 1e6;

}

void Spin (int howlong)

{

double t = GetTime ();

while ((GetTime () - t) < (double) howlong)

; // do nothing in loop

}

unsigned int usecs;

int Car\_Arrive\_N\_N= 0;//this means a car is already present in the signal North 0 - no car, 1 - car present

int Car\_Arrive\_S\_S =0;//this means a car is already present in the signal south 0 - no car, 1 - car present

int Car\_Arrive\_E\_E =0;//this means a car is already present in the signal east 0 - no car, 1 - car present

int Car\_Arrive\_W\_W =0;//this means a car is already present in the signal west 0 - no car, 1 - car present

int counter = 0;

pthread\_mutex\_t linehold\_mutex;// linehold mutex for car in North

pthread\_mutex\_t lineholdS\_mutex;// linehold mutex for car in South

pthread\_mutex\_t lineholdE\_mutex;// linehold mutex for car in East

pthread\_mutex\_t lineholdW\_mutex;// linehold mutex for car in West

pthread\_mutex\_t N\_mutex;//mutex lock for car going from North to north

pthread\_mutex\_t N\_W\_mutex; //mutex lock for car going from North to west

pthread\_mutex\_t S\_mutex; //mutex lock for car going from South to south

pthread\_mutex\_t E\_mutex; //mutex lock for car going from East to east

pthread\_mutex\_t W\_mutex; //mutex lock for car going from West to west

pthread\_cond\_t linehold\_cv;// condition variable to signal car is gone for North(original direction)

pthread\_cond\_t lineholdS\_cv;// condition variable to signal car is gone for Nsouth(original direction)

pthread\_cond\_t lineholdE\_cv;// condition variable to signal car is gone for East(original direction)

pthread\_cond\_t lineholdW\_cv;// condition variable to signal car is gone for west(original direction)

pthread\_cond\_t N\_W\_cv;//CV to signal car(original dir- North, Target dir- West) has passed

pthread\_cond\_t N\_N\_cv ;//CV to signal car(original dir- North, Target dir- North) has passed

typedef struct \_directions

{

char dir\_original;

int car\_id;

char dir\_target;

}directions;

int local\_time=0,local\_time1=0,local\_time2=0,local\_time3=0, local\_time5=0;//variables to output time

int N\_N,N\_W,S\_S,E\_N,W\_N;//if N\_N= 1-> car crossing from North to north

int N\_G=0,N\_Y=0,N\_R=0,S\_G=0,S\_R=0,S\_Y=0;//Green ,red,Yelllow lights for Norts and South

int E\_G=0,E\_Y=0,E\_R=0,W\_G=0,W\_Y=0,W\_R=0; //Green ,red,Yelllow lights for East & West

void ArriveIntersection(void \*arg);

void CrossIntersection(void \*arg) ;

void ExitIntersection(void \*arg) ;

void \*traffic\_signal() {

N\_G=1;S\_G=1;E\_R=1;W\_R=1;//initial values of traffic signal for 1st 18 sec

usleep(18000000);//delay of 18 sec

local\_time2= local\_time2+18000000;

N\_G=0;S\_G=0;//after 18sec , green lights are off and yellow is ON

N\_Y=1;S\_Y=1;

usleep(2000000);//delay

local\_time2= local\_time2+2000000;

//after another 2 sec, East and West become Green, and the corresponding threads are

//signalled

E\_G=1;W\_G=1;N\_R=1;S\_R=1;N\_Y=0;S\_Y=0;

if(E\_G==1){

pthread\_mutex\_lock(&lineholdE\_mutex);

E\_G=1;E\_R=0;

pthread\_cond\_signal(&lineholdE\_cv);

pthread\_mutex\_unlock(&lineholdE\_mutex);

}

}

void \*car(void \*arg) {

directions \*m = (directions \*) arg;

ArriveIntersection(arg);

CrossIntersection(arg);

ExitIntersection(arg);

}

void ArriveIntersection(void \*arg) {

directions \*m = (directions \*) arg;//unpack - car ID, and directions

printf("Time %.2f:Car %d (->%c->%c ) arriving\n ",usecs/1000000.0,m->car\_id,m->dir\_original, m->dir\_target);

if(m->dir\_original == 'N'){

pthread\_mutex\_lock(&linehold\_mutex);

while(Car\_Arrive\_N\_N==1 || N\_R ==1){//car ahead of you or red signal

pthread\_cond\_wait(&linehold\_cv, &linehold\_mutex);}

pthread\_mutex\_unlock(&linehold\_mutex);

}

if(m->dir\_original == 'S' && m->dir\_target== 'S'){

pthread\_mutex\_lock(&lineholdS\_mutex);

while(Car\_Arrive\_S\_S==1 || S\_R ==1){//car ahead of you or red signal

pthread\_cond\_wait(&lineholdS\_cv, &lineholdS\_mutex);}

pthread\_mutex\_unlock(&lineholdS\_mutex);

pthread\_mutex\_lock(&N\_W\_mutex);

while(N\_W==1 || S\_R ==1){//N\_W = 1 means a car is crossing from North to West..so this one should //wait

pthread\_cond\_wait(&N\_W\_cv, &N\_W\_mutex);}

pthread\_mutex\_unlock(&N\_W\_mutex);

}

if(m->dir\_original == 'S' && m->dir\_target== 'E'){

pthread\_mutex\_lock(&N\_W\_mutex);

pthread\_mutex\_lock(&N\_mutex);

while(N\_W==1 || S\_R ==1 || N\_N ==1){//conditions for wait - Car going in North to West direc;

//Car going from North to Norrth

pthread\_cond\_wait(&N\_W\_cv, &N\_W\_mutex);

pthread\_cond\_wait(&N\_N\_cv, &N\_mutex);

}

pthread\_mutex\_unlock(&N\_mutex);

pthread\_mutex\_unlock(&N\_W\_mutex);

}

if(m->dir\_original == 'E' && m->dir\_target== 'N'){

pthread\_mutex\_lock(&lineholdE\_mutex);

while(Car\_Arrive\_E\_E==1 || E\_R ==1){//condition for wait - car ahead of you or red signal

pthread\_cond\_wait(&lineholdE\_cv, &lineholdE\_mutex);}

pthread\_mutex\_unlock(&lineholdE\_mutex);

}

if(m->dir\_original == 'W' && m->dir\_target== 'N'){//line hold lock

pthread\_mutex\_lock(&lineholdW\_mutex);

while(Car\_Arrive\_W\_W==1){//car ahead of you or red signal

pthread\_cond\_wait(&lineholdW\_cv, &lineholdW\_mutex);}

pthread\_mutex\_unlock(&lineholdW\_mutex);

pthread\_mutex\_lock(&W\_mutex);

while(N\_N==1){//N\_N = 1 means a car is crossing from North to North..so this one should wait

pthread\_cond\_wait(&N\_N\_cv, &W\_mutex);}

pthread\_mutex\_unlock(&W\_mutex);

}

}

void CrossIntersection(void \*arg) {

// double time = counter/10.0;

directions \*m = (directions \*) arg;//unpack

if(m->dir\_original=='N'){

pthread\_mutex\_lock(&linehold\_mutex);

Car\_Arrive\_N\_N=0;//release the linehold lock

pthread\_cond\_signal(&linehold\_cv);

pthread\_mutex\_unlock(&linehold\_mutex);

printf("Time %.2f:Car %d (->%c -> %c ) crossing\n ",usecs/1000000.0,m->car\_id,m->dir\_original, m->dir\_target);

}

if(m->dir\_target=='N' && m->dir\_original=='N'){

pthread\_mutex\_lock(&N\_mutex);

N\_N=1;

local\_time= usecs+2000000;//time to simulate crossing

Spin(1);//delay

}

if(m->dir\_target=='W' && m->dir\_original=='N'){

pthread\_mutex\_lock(&N\_W\_mutex);//N\_W

N\_W=1;

int local\_time = usecs +3000000;

Spin(1);

Spin(1);

Spin(1);

}

if(m->dir\_original=='S'){

pthread\_mutex\_lock(&lineholdS\_mutex);

Car\_Arrive\_S\_S=0;

pthread\_cond\_signal(&lineholdS\_cv);

pthread\_mutex\_unlock(&lineholdS\_mutex);

printf("Time %.2f:Car %d (->%c -> %c ) crossing\n ",local\_time/1000000.0,m->car\_id,m->dir\_original, m->dir\_target);

//after above conditions are fulfilled start crossing

pthread\_mutex\_lock(&S\_mutex);

S\_S=1;//started crossing

local\_time5= local\_time;

local\_time= local\_time+2000000;

Spin(1);

}

if(m->dir\_target=='E' && m->dir\_original=='S'){

pthread\_mutex\_lock(&N\_W\_mutex);//N\_W

local\_time5 = local\_time5 +3000000;

Spin(1);//chech spin 2

Spin(1);

}

if(m->dir\_original=='E'){

pthread\_mutex\_lock(&lineholdE\_mutex);

Car\_Arrive\_E\_E=0;

pthread\_cond\_signal(&lineholdE\_cv);

pthread\_mutex\_unlock(&lineholdE\_mutex);

printf("Time %.2f:Car %d (->%c -> %c ) crossing\n ",local\_time2/1000000.0,m->car\_id,m->dir\_original, m->dir\_target); //modify this tomo

pthread\_mutex\_lock(&E\_mutex);//N\_W

E\_N=1;

local\_time2 = local\_time2 +3000000;

Spin(1);//chech spin 2

Spin(1);

Spin(1);

}

if(m->dir\_original == 'W' && m->dir\_target== 'N'){

printf("Time %.2f:Car %d (%c - %c ) crossing3\n ",local\_time/1000000.0,m->car\_id,m->dir\_original, m->dir\_target);

pthread\_mutex\_lock(&W\_mutex);//N\_W

W\_N=1;

local\_time3 = local\_time +1000000;//right turn - 1sec

Spin(1);

}

}

void ExitIntersection(void \*arg) {

directions \*m = (directions \*) arg;

if(m->dir\_target=='N' && m->dir\_original=='N'){

printf("Time %.2f:Car %d (%c - %c ) exit\n",local\_time/1000000.0,m->car\_id,m->dir\_original, m->dir\_target);

N\_N=0;

Spin(1);

pthread\_cond\_signal(&N\_N\_cv);

pthread\_mutex\_unlock(&N\_mutex);

}

if(m->dir\_target=='S' && m->dir\_original=='S'){

// local\_time = usecs +2000000;

printf("Time %.2f:Car %d (->%c -> %c ) exit\n",(local\_time)/1000000.0,m->car\_id,m->dir\_original, m->dir\_target);

S\_S=0;

Spin(1);

pthread\_mutex\_unlock(&S\_mutex);

}

if(m->dir\_target=='W' && m->dir\_original=='N'){

//int local\_time = usecs +3000000;

printf("Time %.2f:Car %d (->%c -> %c ) exit\n",(local\_time)/1000000.0,m->car\_id,m->dir\_original, m->dir\_target);

N\_W=0;

Spin(1);

pthread\_cond\_signal(&N\_W\_cv);

pthread\_mutex\_unlock(&N\_W\_mutex);

}

if(m->dir\_target=='E' && m->dir\_original=='S'){

printf("Time %.2f:Car %d (->%c -> %c ) exit\n",(local\_time5)/1000000.0,m->car\_id,m->dir\_original, m->dir\_target);

Spin(1);

// pthread\_cond\_signal(&N\_W\_cv);

pthread\_mutex\_unlock(&N\_W\_mutex);

}

if(m->dir\_original=='E'){

printf("Time %.2f:Car %d (->%c -> %c ) exit\n",local\_time2/1000000.0,m->car\_id,m->dir\_original, m->dir\_target);

E\_N=0;

Spin(1);

pthread\_mutex\_unlock(&E\_mutex);

}

if(m->dir\_original == 'W' && m->dir\_target== 'N'){

printf("Time %.2f:Car %d (->%c -> %c ) exit\n",local\_time3/1000000.0,m->car\_id,m->dir\_original, m->dir\_target);

W\_N=0;

Spin(1);

pthread\_mutex\_unlock(&W\_mutex);

}

}

int main ()

{

pthread\_t a,t0,t1,t2,t3,t4,t5,t6,t7;

directions dir1,dir2,dir3,dir4,dir5,dir6,dir7,dir8;

usecs =0;//time variable

pthread\_create(&a, NULL, traffic\_signal, NULL);//create thread for traffic signal at time0

usleep(100000);

usecs =500000;

usleep (usecs);

usecs = usecs+600000;

usleep(usecs);

dir1.dir\_original = 'N';

dir1.dir\_target = 'N';

dir1.car\_id = 0;

dir2.dir\_original = 'N';

dir2.dir\_target = 'N';

dir2.car\_id = 1;

dir3.dir\_original = 'N';

dir3.dir\_target = 'W';

dir3.car\_id = 2;

dir4.dir\_original = 'S';

dir4.dir\_target = 'S';

dir4.car\_id = 3;

dir5.dir\_original = 'S';

dir5.dir\_target = 'E';

dir5.car\_id = 4;

dir6.dir\_original = 'N';

dir6.dir\_target = 'N';

dir6.car\_id = 5;

dir7.dir\_original = 'E';

dir7.dir\_target = 'N';

dir7.car\_id = 6;

dir8.dir\_original = 'W';

dir8.dir\_target = 'N';

dir8.car\_id = 7;

pthread\_create(&t0, NULL, car, &dir1);//thread for Car0 at time 1.1

usleep (900000);

usecs= usecs + 900000;

pthread\_create(&t1, NULL, car, &dir2);//thread for Car1 at time 2

usleep (1300000);

usecs= usecs + 1300000;

pthread\_create(&t2, NULL, car, &dir3); //thread for Car2 at time 3.3

usleep(200000);

usecs= usecs +200000;

pthread\_create(&t3, NULL, car, &dir4);//thread for Car3 at time 3.5

usleep(700000);

usecs= usecs + 700000;

pthread\_create(&t4, NULL, car, &dir5);//thread for Car4 at time 4.2

usleep(200000);

usecs = usecs + 200000;

pthread\_create(&t5, NULL, car, &dir6);//thread for Car5 at time 4.4

usleep(1300000);

usecs = usecs + 1300000;

pthread\_create(&t6, NULL, car, &dir7);//thread for Car6 at time 5.7

usleep(200000);

usecs = usecs+200000;

pthread\_create(&t7, NULL, car, &dir8);//thread for Car7 at time 5.9

usleep(27000000);//wait

printf ("End of prog");

return 0;

}