1. --modificat

#include <stdio.h>

#include <unistd.h>

#include <errno.h>

#include <stdlib.h>

#include <pthread.h>

#define MAX\_RESOURCES 5

int available\_resources = MAX\_RESOURCES ;

pthread\_mutex\_t mtx;

int cnt[5]={1,2,3,4,5};

void\* my\_thread(void\* x){

int count = \*((int\*)x);

if(decrease\_count(count)!=-1)

increase\_count(count);

return NULL;

}

int decrease\_count (int count )

{

pthread\_mutex\_lock (& mtx );

if ( available\_resources < count ){

pthread\_mutex\_unlock(&mtx);

return -1;

}

else{

available\_resources -= count ;

printf("Got %d resources %d remaining\n",count, available\_resources);

pthread\_mutex\_unlock (& mtx );

return 0;

}

}

int increase\_count (int count )

{

pthread\_mutex\_lock (& mtx );

available\_resources += count ;

printf("Released %d resources %d remaining\n",count, available\_resources);

pthread\_mutex\_unlock (& mtx );

return 0;

}

int main(){

if (pthread\_mutex\_init(&mtx, NULL)) {

perror(NULL);

return errno ;

}

pthread\_t\* thr = malloc(5 \* sizeof(pthread\_t));

for(int i=0;i<5;++i)

if( pthread\_create(&thr[i], NULL, my\_thread, &cnt[i])){

perror(NULL);

return errno;

}

for(int i=0;i<5;++i)

if( pthread\_join(thr[i], NULL)){

perror(NULL);

return errno;

}

return 0;

}

2.

#include <stdio.h>

#include <unistd.h>

#include <errno.h>

#include <semaphore.h>

#include <stdlib.h>

sem\_t sem;

pthread\_mutex\_t mtx;

int nthrs=5;

void barrier\_point(){

int visited=0;

pthread\_mutex\_lock(&mtx);

visited += 1;

if(visited==nthrs){

pthread\_mutex\_unlock(&mtx);

sem\_post(&sem);

return;

}

pthread\_mutex\_unlock(&mtx);

visited-=1;

}

void \* tfun (void \*v)

{

int \*tid = (int \*)v;

printf ("% d reached the barrier \n", \*tid );

barrier\_point ();

printf ("% d passed the barrier \n", \*tid );

free (tid);

return NULL ;

}

int main(){

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

pthread\_t\* threads = malloc(sizeof(pthread\_t)\*nthrs);

int i;

for(i=0;i<nthrs;i++){

int \*args=malloc(sizeof(int));

\*args=i;

pthread\_create(threads+i,NULL,tfun,args);

}

for(i=0;i<nthrs;i++)

pthread\_join(threads[i],NULL);

free(threads);

return 0;

}