#include <pthread.h>

#include <iostream>

#include <string.h>

#include <stdlib.h>

#include <fcntl.h>

static pthread\_mutex\_t bsem;

static pthread\_cond\_t waitTurn = PTHREAD\_COND\_INITIALIZER;

static int turn;

static int nthreads;

static int even;

using namespace std;

void \*print\_in\_reverse\_order\_odd\_then\_even(void \*void\_ptr\_argv)

{

int threadID = \*((int \*) void\_ptr\_argv);

string str;

pthread\_mutex\_lock(&bsem);

while(turn!=threadID)

pthread\_cond\_wait(&waitTurn,&bsem);

pthread\_mutex\_unlock(&bsem);

std::cout << "I am a thread with an " << str<< "ID equal to "<<threadID << std::endl;

pthread\_mutex\_lock(&bsem);

if ((nthreads-1)%2!=0){ //even

turn = nthreads -1;//highest even

str="even";

}

else{ //odd

turn = nthreads -1;//highest even

str="odd";

}

pthread\_cond\_broadcast(&waitTurn);

pthread\_mutex\_unlock(&bsem);

return NULL;

}

//g++ main.cpp -pthread -o main

//./main

int main()

{

std::cin >> nthreads;

pthread\_mutex\_init(&bsem, NULL); // Initialize access to 1

pthread\_t \*tid= new pthread\_t[nthreads];

int \*threadNumber=new int[nthreads];

turn--;

if ((nthreads-1)%2!=0) //even

even = 1;// highest odd

else //odd

even = 0;//highest even

//Determine the starting thread (thread with the highest odd number)

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

{

threadNumber[i]=i;

pthread\_create(&tid[i],nullptr,print\_in\_reverse\_order\_odd\_then\_even,(void \*) &threadNumber[i]);

}

// Wait for the other threads to finish.

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

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

delete [] threadNumber;

delete [] tid;

return 0;

}