**Código de la practica**

#include <iostream>

#include <stdio.h>

#include <stdlib.h>

#include <signal.h>

#include <string.h>

#include <signal.h>

#include <unistd.h>

#include <windows.h>

#include"lomo2.h"

typedef int (\*LOMO\_generar\_mapa)(char const \*login1, char const \*login2);

typedef int (\*LOMO\_inicio)(int ret, int longitudMAxima, char const \*login1, char const \*login2);

typedef int (\*LOMO\_trenNuevo)(void);

typedef int (\*LOMO\_peticiOnAvance)(int nt, int \*xcab, int \*ycab);

typedef int (\*LOMO\_avance)(int nt, int \*xcola, int \*ycola);

typedef char\* (\*LOMO\_getColor)(int nt);

typedef void (\*LOMO\_espera)(int y, int yn);

typedef int (\*LOMO\_fin)(void);

typedef void (\*pon\_error)(char \*mensaje);

DWORD WINAPI FuncTrenes(LPVOID biblioteca);

BOOL CtrlHandler(DWORD fdwCtrlType);

typedef struct posicion{

int x;

int y;}posicion;

LOMO\_inicio inicio;

LOMO\_trenNuevo trenNuevo;

LOMO\_peticiOnAvance peticion\_avance;

LOMO\_avance avance;

LOMO\_getColor getColor;

LOMO\_espera espera;

LOMO\_fin fin;

pon\_error error;

int nTrenes,velocidad,tamMaximo;

HANDLE semaforos[75][17];

posicion direccionSolicitada[100];

int posTrenx[100];

int posTreny[100];

int main(int argc, char \*argv[]) {

int i,j,x,y;

LOMO\_generar\_mapa generar\_mapa;

HANDLE hilo[100];

HINSTANCE biblioteca;

if( SetConsoleCtrlHandler( (PHANDLER\_ROUTINE) CtrlHandler, TRUE )==FALSE )

{

PERROR("Error al registrar control C");

return 1;

}

if(argc!=2 && argc!=4)

{

fprintf(stderr, "ERROR,ARGUMENTOS NO VALIDOS\n");

return 1;

}

/\*for(i=0; i<100; i++){

direccionSolicitada[i].x=-1 LIBRE;

direccionSolicitada[i].y=-1;

posTrenx[i]=-1;

posTreny[i]=-1;

}Ponemos todas las casillas a libre\*/

biblioteca = LoadLibrary ("lomo2.dll");

if(biblioteca == NULL)

{

fprintf(stderr, "ERROR AL CARGAR BIBLIOTECA\n");

return 1;

}

else

{

generar\_mapa =(LOMO\_generar\_mapa) GetProcAddress(biblioteca,"LOMO\_generar\_mapa");

if(generar\_mapa == NULL)

{

PERROR ("Error GetProcAddress LOMO\_generar\_mapa.");

return 1;

}

inicio=(LOMO\_inicio)GetProcAddress(biblioteca,"LOMO\_inicio");

if(inicio == NULL)

{

PERROR ("Error GetProcAddress LOMO\_inicio.");

return 1;

}

trenNuevo=(LOMO\_trenNuevo)GetProcAddress(biblioteca,"LOMO\_trenNuevo");

if(trenNuevo == NULL)

{

PERROR ("Error GetProcAddress LOMO\_trenNuevo.");

return 1;

}

peticion\_avance=(LOMO\_peticiOnAvance)GetProcAddress(biblioteca,"LOMO\_peticiOnAvance");

if(peticion\_avance == NULL)

{

PERROR ("Error GetProcAddress LOMO\_peticiOnAvance.");

return 1;

}

avance=(LOMO\_avance)GetProcAddress(biblioteca,"LOMO\_avance");

if(avance == NULL)

{

PERROR ("Error GetProcAddress LOMO\_avance.");

return 1;

}

getColor=(LOMO\_getColor)GetProcAddress(biblioteca,"LOMO\_getColor");

if(getColor == NULL)

{

PERROR ("Error GetProcAddress LOMO\_getColor.");

return 1;

}

espera=(LOMO\_espera)GetProcAddress(biblioteca,"LOMO\_espera");

if(espera == NULL)

{

PERROR ("Error GetProcAddress LOMO\_espera.");

return 1;

}

fin=(LOMO\_fin)GetProcAddress(biblioteca,"LOMO\_fin");

if(fin == NULL)

{

PERROR ("Error GetProcAddress LOMO\_fin.");

return 1;

}

error=(pon\_error) GetProcAddress(biblioteca,"pon\_error");

if(error == NULL)

{

PERROR ("Error GetProcAddress pon\_error.");

return 1;

}

}

if(argc==2)

{

if(strcmp(argv[1], "--mapa")==0)

{

//Lamar funcion mapa

generar\_mapa("i6051239","i6040194");

return 0;

}

else

{

fprintf(stderr, "ERROR,NO ESTAMOS EN EL MAPA\n");

return 1;

}

}

if(argc==4)

{

velocidad=atoi(argv[1]);

if(velocidad<0)

{

fprintf(stderr, "Velocidad negativa!\n");

return 1;

}

tamMaximo=atoi(argv[2]);

if(tamMaximo>19)

{

fprintf(stderr, "Tren demasiado grande Max 19!\n");

return 1;

}

nTrenes=atoi(argv[3]);

if(nTrenes<1 || nTrenes >99)

{

fprintf(stderr, "Numero de trenes fuera de rango\n");

return 1;

}

}

inicio(velocidad,tamMaximo,"i6051239","i6040194");

//Creacion de semaforos

for(x=0;x<75;x++)

{

for(y=0;y<17;y++)

{

semaforos[x][y] = CreateSemaphore(NULL, 1, 1, NULL);

if (semaforos[x][y] == NULL)

{

PERROR("Error al crear semaforo control");

exit(1);

}

}

}

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

{

//Creamos un hilo

if( (CreateThread(NULL, 0, FuncTrenes, biblioteca, 0, NULL)) == NULL)

{

PERROR("CreateThread");

return 1;

}

}

Sleep(60000);

if(FreeLibrary(biblioteca) == FALSE)

{

PERROR("Error al liberar la biblioteca\n");

}

return 0;

}

DWORD WINAPI FuncTrenes(LPVOID biblioteca) {

int tren,xcab,ycab,xcola,ycola;

int y=0, ny=0;

int i;

if(WaitForSingleObject(semaforos[0][0],INFINITE) == WAIT\_FAILED)

{

PERROR("Error en el wait del semaforo\n");

exit(10);

}

tren = trenNuevo();

for(;;)

{

peticion\_avance(tren, &xcab, &ycab);

//direccionSolicitada[tren].x = xcab;

//direccionSolicitada[tren].y = ycab;

//detectar\_interbloqueo(tren);

//Aqui hacer waits

if(WaitForSingleObject(semaforos[xcab][ycab],INFINITE) == WAIT\_FAILED)

{

PERROR("Error en el wait del semaforo\n");

exit(10);

}

//posTrenx[xcab]=tren;

//posTreny[ycab]=tren;

//direccionSolicitada[tren].x = -1; //LIBRE

//direccionSolicitada[tren].y= -1;

y= ny;

ny=ycab;

avance(tren, &xcola, &ycola);

//posTrenx[xcola]= -1;

//posTreny[ycola]= -1;

//Aqui hacer signals

if(xcola != -1)

{

if(ReleaseSemaphore(semaforos[xcola][ycola],1,NULL) == FALSE)

{

PERROR("Error en el signal del semaforo\n");

exit(11);

}

}

espera(y,ny);

}

return 0;

}

int detectar\_interbloqueo(int tren)

{

int i, j;

int sigx, sigy, sigTren;

int trenesBloqueados[20];

int numTrenesBloq=0;

char cad[100];

int libre;

sigx=direccionSolicitada[tren].x;

sigy=direccionSolicitada[tren].y;

trenesBloqueados[0]=tren;

numTrenesBloq++;

libre=0;

while((sigx!=-1)&&(sigy!=-1)&&(libre==0)){

libre=1;

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

if((posTrenx[i]==sigx)&&(posTreny[i]==sigy)&&(i!=tren)){

sigTren=i;

sigx=direccionSolicitada[sigTren].x;

sigy=direccionSolicitada[sigTren].y;

trenesBloqueados[numTrenesBloq]=sigTren;

numTrenesBloq++;

libre=0;

if(sigTren==trenesBloqueados[j]){

strcpy(cad, "INTERBLOQUEO ");

for(i=0; i<numTrenesBloq-1; i++){

strcat(cad, getColor(trenesBloqueados[i]));

strcat(cad, " ");

}

error(cad);

exit(0);

}

}

}

}

}

BOOL CtrlHandler(DWORD fdwCtrlType)

{

switch( fdwCtrlType )

{

case CTRL\_C\_EVENT:

fin();

exit(0);

}

}

**ESQUEMA DE SINCRONIZACIÓN**

Para la sincronización de los trenes hemos creado una variable de tipo HANDLE llamada semaforos.

**HANDLE semaforos[75][17];**

Se trata de un array en el que guardamos todas las posiciones de nuestro mapa, controlando así cada una de las coordenadas a las que pueden acceder los trenes.

Primero controlamos la posición (0,0) haciendo un wait sobre el primer semáforo con la función WaitForSingleObject y una vez controlado esto, con la ayuda de un bucle infinito, controlamos las demás posiciones del mapa.

Haciendo un wait para saber a que posición quiere acceder la cabeza del tren y un signal para saber que posición libera la cola del tren

Por\_siempre\_jamas

w(tren)

avanza(tren)

s(tren)