Segunda práctica
CRUCE 2

Primer miembro: ROBERTO MERCHÁN GONZÁLEZ (i0909939)

Segundo miembro: SERGIO SÁNCHEZ GARCÍA (i0961594)

27 de mayo de 2021

Roberto Merchán González i0909939 (robertomergon@usal.es)

Sergio Sánchez García i0961594 (sergiosg@usal.es)

SEGUNDA PRÁCTICA EVALUABLE: CRUCE2

MUTEX

El mutex sC1, lo inicializamos a FALSE. Este mutex controla el semáforo C1 para evitar que los coches que aparecen por la carretera vertical no puedan cruzar la línea de detención cuando C1 está en rojo o amarillo.

```
ciclosem{
        W(sC1)
        pon color amarillo
        pon color rojo
```

```
esCoche{
        si pos.x==33 Y pos.y=6 entonces
                W(sC1)
        avanza el coche
```

El mutex sC2, lo inicializamos a FALSE. Este mutex controla el semáforo C2 para evitar que los coches que aparecen por la carretera horizontal no puedan cruzar la línea de detención cuando C2 está en rojo o amarillo.

```
ciclosem{
        R(sC2)
        pon color verde
        W(sC2)
```

```
esCoche{
        si pos.x==13 Y pos.y=10 entonces
                W(sC2)
        avanza el coche
```

El mutex sP1, lo inicializamos a FALSE. Este mutex controla el semáforo P1 para evitar que los peatones no puedan cruzar por el paso de cebra horizontal cuando P1 está en rojo.

```
ciclosem{
        pon color verde
        R(sC1)
        W(sC1)
```

```
esPeaton{
        si pos.x==30 Y pos.y<=15 Y pos.y>=13 entonces
                W(sP1)
        avanza el peatón
```

Roberto Merchán González i0909939 (robertomergon@usal.es)

Sergio Sánchez García i0961594 (sergiosg@usal.es)

El mutex sP2, lo inicializamos a FALSE. Este mutex controla el semáforo P2 para evitar que los peatones no puedan cruzar por el paso de cebra vertical cuando P2 está en rojo.

```
ciclosem{
        W(sP2)
        pon color rojo
        pon color verde
```

```
esPeaton{
        si pos.y==11 Y pos.x<=28 Y pos.x>=22 entonces
                W(sP2)
        avanza el peatón
```

El mutex sAmarillo, lo inicializamos a FALSE. Este mutex controla el tiempo que debe durar el color amarillo en los semáforos de los coches.

```
ciclosem{
        pon color amarillo C1
        W(sAmarillo)
        pausas
        R(sAmarillo)
        pon color rojo C1
        pon color verde C2
        pon color amarillo C2
```

```
esCoche{
        si pos.x==33 Y pos.y=6 entonces
                W(sAmarillo)
        si pos.x==13 Y pos.y=10 entonces
                W(sAmarillo)
        avanza el coche
        si pos.y<0 entonces
                R(sAmarillo)
```

El mutex sNacPeaton, lo inicializamos a FALSE. Este se encarga de que no pueda nacer un nuevo peatón mientras otro este en la zona crítica en la que pueden nacer.

```
esPeaton{
        W(sNacPeaton)
        Si NO esta en zona crítica de nacimiento peatones entonces
                R(sNacPeaton)
}
```

Roberto Merchán González i0909939 (robertomergon@usal.es)

Sergio Sánchez García i0961594 (sergiosg@usal.es)

SECCION CRÍTICA

Utilizamos una sección crítica, scCruce, para controlar que en un momento dado solo pueda entrar un coche en el cruce. Cuando un coche entra en el cruce, entra en la sección crítica y no sale de esta hasta que salga del cruce.

```
esCoche{
        si pos.x==33 Y pos.y=6 entonces
                enter(scCruce)
        si pos.x==13 Y pos.y=10 entonces
                enter(scCruce)
```

SEMÁFOROS

El semáforo sMaxNumProc, lo inicializamos al número de hilos máximos introducidos por la línea de ordenes menos dos debido al proceso padre y al hilo encargado de controlar el ciclo semafórico. Este es el encargado de controlar el número de hilos existentes en el mapa.

```
main{
        si creacion==COCHE entonces
                W(sMaxNumProc)
                crearHiloCoche
        Si creacion==PEATON entonces
                W(sMaxNumProc)
                crearHiloPeaton
}
```

```
esCoche{
        si pos.y<0 entonces
                R(smaxNumProc)
}
```

```
esPeaton{
        si pos.y<0 entonces
                R(smaxNumProc)
}
```

Roberto Merchán González i0909939 (robertomergon@usal.es)

Sergio Sánchez García i0961594 (sergiosg@usal.es)

CONTROL DE CHOQUES

Utilizamos un vector de mutex, sSeguridadVialH de dimensión 18 para controlar el choque de los coches que circulan por la carretera horizontal.
El coche, antes de avanzar espera hasta que queda libre el mutex correspondiente a la posición la que va a avanzar. Una vez avanza, libera la posición_anterior_x

```
esCoche{

        x = 1

        si NO nacimientoV Y pos.x < 13 entonces

                W(sSeguridadVialH[x + 4])

                        x += 2

        si NO nacimientoV Y pos.y == 10 Y pos.x > 5 entonces

                R(sSeguridadVialH[pos_ant.x])

}
```

Utilizamos un vector de mutex, sSeguridadVialV de dimensión 7 para controlar el choque de los coches que circulan por la carretera horizontal.
El coche, antes de avanzar espera hasta que queda libre el mutex correspondiente a la posición la que va a avanzar. Una vez avanza, libera la posición_anterior_y

```
esCoche{

        y = 0

        si  nacimientoV Y pos.y < 7 entonces

                W(sSeguridadVialV[y])

                y += 1

        si NO nacimientoV Y pos.x == 33 Y pos.y < 7 entonces

                R(sSeguridadVialV[pos_ant.y])

}
```

Roberto Merchán González i0909939 (robertomergon@usal.es)

Sergio Sánchez García i0961594 (sergiosg@usal.es)

Utilizamos un array de mutex, sChoquePeaton, para controlar el choque de los peatones. Antes de avanzar a una posción, el peatón espera a que el mutex asociado a esta esté libre. Una vez avanza, libera la posición en la que se encontraba previamente.

```
esPeaton{
        W(sChoquePeaton[pos.x][pos.y])
        pos_ant = pos
        avanza el peaton
        si primerMov entonces
                pomerMov = 0
```

Roberto Merchán González i0909939 (robertomergon@usal.es)

Sergio Sánchez García i0961594 (sergiosg@usal.es)

CÓDIGO

```
//  ROBERTO MERCHAN GONZALEZ robertomergon@usal.es
//  SERGIO SANCHEZ GARCIA    sergiosg@usal.es

#include "Windows.h"
#include "iostream"
#include "stdio.h"
#include "stdlib.h"
#include "cruce2.h"
#include "signal.h"

#define POSIC_x 18
#define POSIC_y  7
#define ANCHO 50
#define ALTO 17
#define MAX_PROCESOS 49
#define MIN_PROCESOS 3
#define VELOCIDAD_MIN 0

// Prototipos funciones
BOOL WINAPI handler(DWORD fdwCtlrType);
int liberar(void);
void despedida(void);
DWORD WINAPI cicloSem(LPVOID);
DWORD WINAPI esCoche(LPVOID);
DWORD WINAPI esPeaton(LPVOID);
int mapeado();
int reservaIPCS(int);
// Protitipos DLL
int (*dllini) (int, int);
int (*dllCambiarColor) (int, int);
struct posiciOn(*dllAvanzaCoche) (struct posiciOn);
struct posiciOn(*dllAvanzaPeaton) (struct posiciOn);
void (*dllPonError) (const char*);
struct posiciOn(*dllIniCoche) (void);
struct posiciOn(*dllIniPeaton) (void);

// Variables globales
HINSTANCE libreria;
FARPROC dllFin, dllGestor, dllNuevoProc, dllFinCoche, dllFinPeaton, dllPausa,
dllPausaCoche, dllRefrescar;
HANDLE sC1, sC2, sP1, sP2, sAmarillo, sMaxNumProc, sNacPeaton;
```

Roberto Merchán González i0909939 (robertomergon@usal.es)

Sergio Sánchez García i0961594 (sergiosg@usal.es)

```
HANDLE sSeguridadVialV[POSIC_y], sSeguridadVialH[POSIC_x],
sChoquePeaton[ALTO][ANCHO];
CRITICAL_SECTION scCruce;
HANDLE threadC, threadP, threadCicloSem;

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

	if (!SetConsoleCtrlHandler(handler, TRUE)) {
		fprintf(stderr, "ERROR : CTLR+C");
		exit(1);
	}
	// Comprobación de parámetros
	if (argc != 3) {
		fprintf(stderr, "Error en el numero de parametros por la linea de
ordenes\n");
		exit(2);
	}
	if (atoi(argv[1]) < MIN_PROCESOS || atoi(argv[1]) > MAX_PROCESOS ||
atoi(argv[2]) < VELOCIDAD_MIN) {
		fprintf(stderr, "Error en los valores del parametro\n");
		exit(3);
	}
	// Carga DLL
	libreria = LoadLibrary("cruce2.dll");
	if (libreria == NULL) {
		fprintf(stderr, "NO SE HA PODIDO CARGAR LA LIBRERIA\n");
		fflush(stdout);
		return -1;
	}
	else fprintf(stderr, "SE HA CARGADO LA LIBRERIA\n");
	if (mapeado() == -1) {
		fprintf(stderr, "Error al mapear");
		return -1;
	}
	if (reservaIPCS(atoi(argv[1])) == -1) {
		fprintf(stderr, "Error al reservar los IPCS");
		return -1;
	}
	// Inicio de CRUCE
	if (dllini(atoi(argv[2]), atoi(argv[1])) == -1) {
		fprintf(stderr, "Error en la funcion CRUCE_inicio");
		return -1;
	}
	threadCicloSem = CreateThread(NULL, 0, cicloSem, LPVOID('a'), 0, NULL);
	if (threadCicloSem == NULL) {
		return -1;
```

```
	}
	for (;;) {
		if ((creacion = dllNuevoProc()) == -1) {
			fprintf(stderr, "Error en la funcion CRUCE_nuevo_proceso");
			return -1;
		}
		else if (creacion == COCHE) {		//Es un coche
			WaitForSingleObject(sMaxNumProc, INFINITE);
			threadC = CreateThread(NULL, 0, esCoche, NULL, 0, NULL);
			if (threadC == NULL) {
				return -1;
			}
		}
		else if (creacion == PEAToN) {  //Es un peaton
			WaitForSingleObject(sMaxNumProc, INFINITE);
			threadP = CreateThread(NULL, 0, esPeaton, NULL, 0, NULL);
			if (threadP == NULL) {
				return -1;
			}
		}
	}

}

// CTLR + C
BOOL WINAPI handler(DWORD fdwCtlrType) {
	switch (fdwCtlrType) {
	case CTRL_C_EVENT:
		exit(liberar());
	}
	return TRUE;
}

int liberar() {
	int i, j;

	dllFin();
	DeleteCriticalSection(&scCruce);
	if (0 == CloseHandle(sC1)) {
		fprintf(stderr, "ERROR : cierre de sC1");
		return -1;
	}
	if (0 == CloseHandle(sC2)) {
		fprintf(stderr, "ERROR : cierre de sC2");
		return -1;
	}
	if (0 == CloseHandle(sP1)) {
```

```
		fprintf(stderr, "ERROR : cierre de sP1");
		return -1;
	}
	if (0 == CloseHandle(sP2)) {
		fprintf(stderr, "ERROR : cierre de sP2");
		return -1;
	}
	if (0 == CloseHandle(sAmarillo)) {
		fprintf(stderr, "ERROR : cierre de sAmarillo");
		return -1;
	}
	if (0 == CloseHandle(sMaxNumProc)) {
		fprintf(stderr, "ERROR : cierre de sMaxNumProc");
		return -1;
	}
	if (0 == CloseHandle(sNacPeaton)) {
		fprintf(stderr, "ERROR : cierre de sNacPeaton");
		return -1;
	}
	if (0 == CloseHandle(threadC)) {
		fprintf(stderr, "ERROR : cierre de threadC");
		return -1;
	}
	if (0 == CloseHandle(threadP)) {
		fprintf(stderr, "ERROR : cierre de threadP");
		return -1;
	}
	for (i = 0; i < POSIC_y; i++) {
		if (0 == CloseHandle(sSeguridadVialV[i])) {
			fprintf(stderr, "ERROR : liberacion de sSeguridadVial[POSIC]");
			return -1;
		}
	}
	for (i = 0; i < POSIC_x; i++) {
		if (0 == CloseHandle(sSeguridadVialH[i])) {
			fprintf(stderr, "ERROR : liberacion de sSeguridadVial[POSIC]");
			return -1;
		}
	}
	for (i = 0; i < ALTO; i++) {
		for (j = 0; j < ANCHO; j++) {
			if (0 == CloseHandle(sChoquePeaton[i][j])) {
				fprintf(stderr, "ERROR : liberacion de sChoquePeaton");
				return -1;
			}
		}
	}
```

```
	CloseHandle(threadCicloSem);
	if (FreeLibrary(libreria) == 0)
		fprintf(stderr, "\nLibreria no liberada ;(\n");
	despedida();
	return 0;
}

void despedida() {
	system("CLS");
	printf("\n\n\n");
	printf("\t+------------------------------------------------+");
	printf("\n\t|             TRABAJO REALIZADO POR              |\n");
	printf("\t+------------------------------------------------+");
	printf("\n\t|       ROBERTO MERCHAN GONZALEZ (i0909939)      |\n");
	printf("\t|       SERGIO  SANCHEZ GARCIA   (i0961594)      |\n");
	printf("\t+------------------------------------------------+");
	printf("\n\n\n\n\n");
}

int reservaIPCS(int argv1) {
	// Variables
	int i, j;

	// Reserva de IPCS
	if (NULL == (sC1 = CreateMutex(NULL, FALSE, NULL))) {
		fprintf(stderr, "ERROR : reserva de sC1");
		return -1;
	}
	if (NULL == (sC2 = CreateMutex(NULL, FALSE, NULL))) {
		fprintf(stderr, "ERROR : reserva de sC2");
		return -1;
	}
	if (NULL == (sP1 = CreateMutex(NULL, FALSE, NULL))) {
		fprintf(stderr, "ERROR : reserva de sP1");
		return -1;
	}
	if (NULL == (sP2 = CreateMutex(NULL, FALSE, NULL))) {
		fprintf(stderr, "ERROR : reserva de sP2");
		return -1;
	}
	if (NULL == (sAmarillo = CreateMutex(NULL, FALSE, NULL))) {
		fprintf(stderr, "ERROR : reserva de sAmarillo");
		return -1;
	}
	if (NULL == (sMaxNumProc = CreateSemaphore(NULL, argv1 - 2, argv1 - 2,
NULL))) {
		fprintf(stderr, "ERROR : reserva de sMaxProcNum");
```

```
		return -1;
	}
	if (NULL == (sNacPeaton = CreateMutex(NULL, FALSE, NULL))) {
		fprintf(stderr, "ERROR : reserva de sMaxProcNum");
		return -1;
	}
	InitializeCriticalSection(&scCruce);
	for (i = 0; i < POSIC_y; i++) {
		if (NULL == (sSeguridadVialV[i] = CreateMutex(NULL, FALSE,
NULL))) {
			fprintf(stderr, "ERROR : reserva de sSeguridadVial[POSIC]");
			return -1;
		}
	}
	for (i = 0; i < POSIC_x; i++) {
		if (NULL == (sSeguridadVialH[i] = CreateMutex(NULL, FALSE,
NULL))) {
			fprintf(stderr, "ERROR : reserva de sSeguridadVial[POSIC]");
			return -1;
		}
	}
	for (i = 0; i < ALTO; i++) {
		for (j = 0; j < ANCHO; j++) {
			if (NULL == (sChoquePeaton[i][j] = CreateMutex(NULL,
FALSE, NULL))) {
				fprintf(stderr, "ERROR : reserva de sChoquePeaton");
				return -1;
			}
		}
	}
	return 0;
}

int mapeado() {
	if (NULL == (dllini = (int (*) (int, int))GetProcAddress(libreria,
"CRUCE_inicio"))) {
		fprintf(stderr, "NO SE HA PODIDO MAPEAR CORRECTAMENTE");
		return -1;
	}
	if (NULL == (dllFin = GetProcAddress(libreria, "CRUCE_fin"))) {
		fprintf(stderr, "NO SE HA PODIDO MAPEAR CORRECTAMENTE");
		return -1;
	}
	if (NULL == (dllGestor = GetProcAddress(libreria, "CRUCE_gestor_inicio"))) {
		fprintf(stderr, "NO SE HA PODIDO MAPEAR CORRECTAMENTE");
		return -1;
	}
```

Roberto Merchán González i0909939 (robertomergon@usal.es)

Sergio Sánchez García i0961594 (sergiosg@usal.es)

```
	if (NULL == (dllCambiarColor = (int (*) (int, int)) GetProcAddress(libreria,
"CRUCE_pon_semAforo"))) {
		fprintf(stderr, "NO SE HA PODIDO MAPEAR CORRECTAMENTE");
		return -1;
	}
	if (NULL == (dllNuevoProc = GetProcAddress(libreria,
"CRUCE_nuevo_proceso"))) {
		fprintf(stderr, "NO SE HA PODIDO MAPEAR CORRECTAMENTE");
		return -1;
	}
	if (NULL == (dllIniCoche = (struct posiciOn(*) (void))GetProcAddress(libreria,
"CRUCE_inicio_coche"))) {
		fprintf(stderr, "NO SE HA PODIDO MAPEAR CORRECTAMENTE");
		return -1;
	}
	if (NULL == (dllAvanzaCoche = (struct posiciOn(*) (struct
posiciOn))GetProcAddress(libreria, "CRUCE_avanzar_coche"))) {
		fprintf(stderr, "NO SE HA PODIDO MAPEAR CORRECTAMENTE");
		return -1;
	}
	if (NULL == (dllFinCoche = GetProcAddress(libreria, "CRUCE_fin_coche"))) {
		fprintf(stderr, "NO SE HA PODIDO MAPEAR CORRECTAMENTE");
		return -1;
	}
	if (NULL == (dllIniPeaton = (struct posiciOn(*) (void))GetProcAddress(libreria,
"CRUCE_nuevo_inicio_peatOn"))) {
		fprintf(stderr, "NO SE HA PODIDO MAPEAR CORRECTAMENTE");
		return -1;
	}
	if (NULL == (dllAvanzaPeaton = (struct posiciOn(*) (struct
posiciOn))GetProcAddress(libreria, "CRUCE_avanzar_peatOn"))) {
		fprintf(stderr, "NO SE HA PODIDO MAPEAR CORRECTAMENTE");
		return -1;
	}
	if (NULL == (dllFinPeaton = GetProcAddress(libreria, "CRUCE_fin_peatOn")))
{
		fprintf(stderr, "NO SE HA PODIDO MAPEAR CORRECTAMENTE");
		return -1;
	}
	if (NULL == (dllPausa = GetProcAddress(libreria, "pausa"))) {
		fprintf(stderr, "NO SE HA PODIDO MAPEAR CORRECTAMENTE");
		return -1;
	}
	if (NULL == (dllPausaCoche = GetProcAddress(libreria, "pausa_coche"))) {
		fprintf(stderr, "NO SE HA PODIDO MAPEAR CORRECTAMENTE");
		return -1;
	}
```

```
	if (NULL == (dllRefrescar = GetProcAddress(libreria, "refrescar"))) {
		fprintf(stderr, "NO SE HA PODIDO MAPEAR CORRECTAMENTE");
		return -1;
	}
	if (NULL == (dllPonError = (void (*) (const char*))GetProcAddress(libreria,
"pon_error"))) {
		fprintf(stderr, "NO SE HA PODIDO MAPEAR CORRECTAMENTE");
		return -1;
	}
	return 0;
}

DWORD WINAPI cicloSem(LPVOID a) {
	int i;

	if (dllGestor() == -1) {
		fprintf(stderr, "Error en la función CRUCE_gestor_inicio");
		return -1;
	}
	for (;;) {
		// Amarillo C1
		WaitForSingleObject(sC1, INFINITE);
		if (dllCambiarColor(SEM_C1, AMARILLO) == -1) {
			fprintf(stderr, "Error en la función CRUCE_pon_semAforo");
			return -1;
		}
		WaitForSingleObject(sAmarillo, INFINITE);
		for (i = 0; i < 2; i++) {
			if (dllPausa() == -1) {
				fprintf(stderr, "Error en la función pausa");
				return -1;
			}
		}
		ReleaseMutex(sAmarillo);
		//SEGUNDA FASE
		if (dllCambiarColor(SEM_C1, ROJO) == -1) {
			fprintf(stderr, "Error en la función CRUCE_pon_semAforo");
			return -1;
		}
		WaitForSingleObject(sP2, INFINITE);
		if (dllCambiarColor(SEM_P2, ROJO) == -1) {
			fprintf(stderr, "Error en la función CRUCE_pon_semAforo");
			return -1;
		}
		if (dllCambiarColor(SEM_C2, VERDE) == -1) {
			fprintf(stderr, "Error en la función CRUCE_pon_semAforo");
			return -1;
```

```
}
ReleaseMutex(sC2);
for (i = 0; i < 9; i++) {
	if (dllPausa() == -1) {
		fprintf(stderr, "Error en la función pausa");
		return -1;
	}
}
// Amarillo C2
WaitForSingleObject(sC2, INFINITE);
if (dllCambiarColor(SEM_C2, AMARILLO) == -1) {
	fprintf(stderr, "Error en la función CRUCE_pon_semAforo");
	return -1;
}
WaitForSingleObject(sAmarillo, INFINITE);
for (i = 0; i < 2; i++) {
	if (dllPausa() == -1) {
		fprintf(stderr, "Error en la función pausa");
		return -1;
	}
}
ReleaseMutex(sAmarillo);
//TERCERA FASE
if (dllCambiarColor(SEM_C2, ROJO) == -1) {
	fprintf(stderr, "Error en la función CRUCE_pon_semAforo");
	return -1;
}
if (dllCambiarColor(SEM_P1, VERDE) == -1) {
	fprintf(stderr, "Error en la función CRUCE_pon_semAforo");
	return -1;
}
ReleaseMutex(sP1);
for (i = 0; i < 12; i++) {
	if (dllPausa() == -1) {
		fprintf(stderr, "Error en la función pausa");
		return -1;
	}
}
//PRIMERA FASE
WaitForSingleObject(sP1, INFINITE);
if (dllCambiarColor(SEM_P1, ROJO) == -1) {
	fprintf(stderr, "Error en la función CRUCE_pon_semAforo");
	return -1;
}
if (dllCambiarColor(SEM_C1, VERDE) == -1) {
	fprintf(stderr, "Error en la función CRUCE_pon_semAforo");
	return -1;
```

```
			}
			ReleaseMutex(sC1);
			if (dllCambiarColor(SEM_P2, VERDE) == -1) {
				fprintf(stderr, "Error en la función CRUCE_pon_semAforo");
				return -1;
			}
			ReleaseMutex(sP2);
			for (i = 0; i < 6; i++) {
				if (dllPausa() == -1) {
					fprintf(stderr, "Error en la función pausa");
					return -1;
				}
			}
		}
}

DWORD WINAPI esCoche(LPVOID a) {
		// Variables
		int nacimientoV = 0, rojo = 0, entraCruce = 0, pasa = 0, x = 1, y = 0;
		struct posiciOn pos, pos_ant;

		// Funcionamiento del coche
		pos = dllIniCoche();
		if (pos.x == 33 && pos.y == 1)
			nacimientoV = 1;
		for (;;) {
			if (pos.x == 33 && pos.y == 6) { // C1 rojo
				WaitForSingleObject(sC1, INFINITE);
				rojo = 1;
				EnterCriticalSection(&scCruce);
				WaitForSingleObject(sAmarillo, INFINITE);
				entraCruce = 1;
			}
			if (pos.x == 13 && pos.y == 10) { // C2 rojo
				WaitForSingleObject(sC2, INFINITE);
				rojo = 2;
				EnterCriticalSection(&scCruce);
				WaitForSingleObject(sAmarillo, INFINITE);
				entraCruce = 1;
			}
			if (nacimientoV && pos.y < 6) {
				WaitForSingleObject(sSeguridadVialV[y], INFINITE);
				y++;
			}
			if (!nacimientoV && pos.x < 13) {
				WaitForSingleObject(sSeguridadVialH[x + 4], INFINITE);
				x += 2;
```

Roberto Merchán González i0909939 (robertomergon@usal.es)

Sergio Sánchez García i0961594 (sergiosg@usal.es)

```
			}
			pos_ant = pos;
			pos = dllAvanzaCoche(pos);
			if (dllPausaCoche() == -1) {
				fprintf(stderr, "Error en la funcion pausa_coche");
				return -1;
			}
			if (rojo == 1 && !pasa) {
				ReleaseMutex(sC1);
				pasa = 1;
			}
			if (rojo == 2 && !pasa) {
				ReleaseMutex(sC2);
				pasa = 1;
			}
			if (nacimientoV && pos.x == 33 && pos.y < 7) {
				ReleaseMutex(sSeguridadVialV[pos_ant.y]);
			}
			if (!nacimientoV && pos.x > 5 && pos.y == 10) {
				ReleaseMutex(sSeguridadVialH[pos_ant.x]);
			}
			if (pos.y < 0) // El coche termina
				break;
		}
		if (dllFinCoche() == -1) {
			fprintf(stderr, "Error en la funcion CRUCE_fin_coche");
			return -1;
		}
		ReleaseMutex(sAmarillo);
		LeaveCriticalSection(&scCruce);	// Decrementamos S6 al salir del cruce
		ReleaseSemaphore(sMaxNumProc, 1, NULL);	// Incrementamos S5 al salir
del mapa
		return 1;
}

DWORD WINAPI esPeaton(LPVOID) {
		// Variables
		struct posiciOn pos, pos_ant, pos_ant2;
		int entraCruce1 = 0, entraCruce2 = 0, flagSem = 1, primerMov = 1;

		// Manejo de peaton
		WaitForSingleObject(sNacPeaton, INFINITE);
		pos = dllIniPeaton();
		for (;;) {
			if (pos.x == 30 && (pos.y <= 15 && pos.y >= 13)) { // P1 rojo
				WaitForSingleObject(sP1, INFINITE);
				entraCruce1 = 1;
```

Roberto Merchán González i0909939 (robertomergon@usal.es)

Sergio Sánchez García i0961594 (sergiosg@usal.es)

```
		}
		if (pos.y == 11 && (pos.x <= 28 && pos.x >= 22)) { // P2 rojo
			WaitForSingleObject(sP2, INFINITE);
			entraCruce2 = 1;
		}
		WaitForSingleObject(sChoquePeaton[pos.y][pos.x], INFINITE);
		pos_ant = pos;
		pos = dllAvanzaPeaton(pos);

		if (primerMov)
			primerMov = 0;
		else
			ReleaseMutex(sChoquePeaton[pos_ant2.y][pos_ant2.x]);
		pos_ant2 = pos_ant;
		if (!((pos_ant.x == 0 && pos_ant.y > 10) || (pos_ant.x < 40 &&
pos_ant.y == 16)) && flagSem) {
			ReleaseMutex(sNacPeaton);
			flagSem = 0;
		}
		if (dllPausa() == -1) {
			fprintf(stderr, "Error en la funcion pausa");
			return -1;
		}
		if (entraCruce1 && pos.x == 38) {   // Vertical
			ReleaseMutex(sP1);
			entraCruce1 = 0;
		}
		if (entraCruce2 && pos.y == 6) {     // Horizontal
			ReleaseMutex(sP2);
			entraCruce2 = 0;
		}
		if (pos.y < 0)   // El peaton termina
			break;
	}
	if (dllFinPeaton() == -1) {
		fprintf(stderr, "Error en la funcion CRUCE_fin_peatOn");
		return -1;
	}
	ReleaseSemaphore(sMaxNumProc, 1, NULL);
	return 1;
}
```