

Carrera de caballos

Generated by Doxygen 1.8.11

Contents

1	Data Structure Index	1
1.1	Data Structures	1
2	File Index	3
2.1	File List	3
3	Data Structure Documentation	5
3.1	Carrera_info Struct Reference	5
3.1.1	Detailed Description	5
3.2	Hilo_args Struct Reference	5
3.2.1	Detailed Description	6
3.3	Msg_apuesta Struct Reference	6
3.3.1	Detailed Description	6
4	File Documentation	7
4.1	carrera.c File Reference	7
4.2	lib.c File Reference	7
4.2.1	Detailed Description	8
4.2.2	Function Documentation	8
4.2.2.1	aleat_num(int inf, int sup)	8
4.2.2.2	apostador(int dinero, int caballos, int apostadores, int id_cola)	8
4.2.2.3	BubbleSort(int *ganadores, double *pago, int ip, int iu)	9
4.2.2.4	caballo(int num, int longitud, int *pipelda, int *pipeVuelta)	9
4.2.2.5	captura(int sennal)	9
4.2.2.6	gestor(sigset_t mask, int id_cola, int caballos, int apostadores, int ventanillas, int id)	9

4.2.2.7	monitor(sigset_t mask, int caballos, int longitud, int semid, Carrera_info *info, int dinero)	10
4.2.2.8	principal(sigset_t mask, int **pipelda, int **pipeVuelta, int caballos, int longitud, int semid, Carrera_info *info, pid_t *childpid)	10
4.2.2.9	ventanilla(Hilo_args *arg)	11
4.3	lib.h File Reference	11
4.3.1	Detailed Description	12
4.3.2	Function Documentation	13
4.3.2.1	aleat_num(int inf, int sup)	13
4.3.2.2	apostador(int dinero, int caballos, int apostadores, int id_cola)	13
4.3.2.3	BubbleSort(int *ganadores, double *pago, int ip, int iu)	13
4.3.2.4	caballo(int num, int longitud, int *pipelda, int *pipeVuelta)	14
4.3.2.5	captura(int sennal)	14
4.3.2.6	gestor(sigset_t mask, int id_cola, int caballos, int apostadores, int ventanillas, int id)	14
4.3.2.7	monitor(sigset_t mask, int caballos, int longitud, int semid, Carrera_info *info, int dinero)	15
4.3.2.8	principal(sigset_t mask, int **pipelda, int **pipeVuelta, int caballos, int longitud, int semid, Carrera_info *info, pid_t *childpid)	15
4.3.2.9	ventanilla(Hilo_args *arg)	15
4.4	semaforos.c File Reference	16
4.4.1	Detailed Description	16
4.4.2	Function Documentation	16
4.4.2.1	Borrar_Semaforo(int semid)	16
4.4.2.2	Crear_Semaforo(key_t key, int size, int *semid)	17
4.4.2.3	Down_Semaforo(int id, int num_sem, int undo)	17
4.4.2.4	DownMultiple_Semaforo(int id, int size, int undo, int *active)	17
4.4.2.5	Inicializar_Semaforo(int semid, unsigned short *array)	18
4.4.2.6	Up_Semaforo(int id, int num_sem, int undo)	18
4.4.2.7	UpMultiple_Semaforo(int id, int size, int undo, int *active)	18
	Index	19

Chapter 1

Data Structure Index

1.1 Data Structures

Here are the data structures with brief descriptions:

Carrera_info	Memoria compartida entre principal, gestor y monitor	5
Hilo_args	Estructura pasada como argumento a los hilos num_ventana : numero de ventanilla que es id_mem : id de la memoria compartida	5
Msg_apuesta	Mensaje de cada apostador al gestor de apuestas	6

Chapter 2

File Index

2.1 File List

Here is a list of all documented files with brief descriptions:

carrera.c	Main de la carrera de caballos	7
lib.c	Libreria para las funciones de la carrera entre las que se encuentran las funciones de cada proceso hijo y padre, hilos y auxiliares	7
lib.h	Libreria para las funciones de la carrera entre las que se encuentran las funciones de cada proceso hijo y padre, hilos y auxiliares	11
semaforos.c	Biblioteca de semaforos	16
semaforos.h	??

Chapter 3

Data Structure Documentation

3.1 Carrera_info Struct Reference

Memoria compartida entre principal, gestor y monitor.

```
#include <lib.h>
```

Collaboration diagram for Carrera_info:

Data Fields

- int **tiradas** [[MAX_CABALLOS](#)]
- int **recorridos** [[MAX_CABALLOS](#)]
- int **acabado**
- double **cotizacion** [[MAX_CABALLOS](#)]
- int **total_apuestas**
- int **total_caballo** [[MAX_CABALLOS](#)]
- int **cont**
- [Msg_apuesta](#) **mensaje** [[MAX_APOSTADORES](#)]

3.1.1 Detailed Description

Memoria compartida entre principal, gestor y monitor.

The documentation for this struct was generated from the following file:

- [lib.h](#)

3.2 Hilo_args Struct Reference

Estructura pasada como argumento a los hilos num_ventana : numero de ventanilla que es id_mem : id de la memoria compartida.

```
#include <lib.h>
```

Data Fields

- int **num_ventana**
- int **id_cola**
- int **id_mem**

3.2.1 Detailed Description

Estructura pasada como argumento a los hilos num_ventana : numero de ventanilla que es id_mem : id de la memoria compartida.

The documentation for this struct was generated from the following file:

- [lib.h](#)

3.3 Msg_apuesta Struct Reference

Mensaje de cada apostador al gestor de apuestas.

```
#include <lib.h>
```

Data Fields

- long **mtype**
- char **nombre** [16]
- double **apuesta**
- int **caballo**
- int **ventanilla**
- double **cotizacion**
- double **pago**

3.3.1 Detailed Description

Mensaje de cada apostador al gestor de apuestas.

The documentation for this struct was generated from the following file:

- [lib.h](#)

Chapter 4

File Documentation

4.1 carrera.c File Reference

main de la carrera de caballos

```
#include "lib.h"
```

Include dependency graph for carrera.c:

4.2 lib.c File Reference

Libreria para las funciones de la carrera entre las que se encuentran las funciones de cada proceso hijo y padre, hilos y auxiliares.

```
#include "lib.h"
```

Include dependency graph for lib.c:

Functions

- int [principal](#) (sigset_t mask, int **pipelda, int **pipeVuelta, int caballos, int longitud, int semid, [Carrera_info](#) *info, pid_t *childpid)
Funcion del proceso padre del programa, encargado de realizar la comunicacion entre procesos y de asignarle tiradas a los caballos.
- int [caballo](#) (int num, int longitud, int *pipelda, int *pipeVuelta)
Funcion que realiza el trabajo de los caballos mandando y recibiendo posiciones y recorridos con el padre (principal)
- int [monitor](#) (sigset_t mask, int caballos, int longitud, int semid, [Carrera_info](#) *info, int dinero)
Funcion encargada de imprimir por pantalla la informacion.
- int [apostador](#) (int dinero, int caballos, int apostadores, int id_cola)
Funcion encargada de crear apostadores y mandar mensajes a las ventanillas.
- int [gestor](#) (sigset_t mask, int id_cola, int caballos, int apostadores, int ventanillas, int id)
Funcion encargada de inicializar las variables de la memoria compartida relacionadas con el dinero y de crear las ventanillas.
- void * [ventanilla](#) ([Hilo_args](#) *arg)
Funcion de los hilos encargada de recibir las apuestas y agregarlas a la memoria compartida.
- void [captura](#) (int sennal)
Funcion capturadora de las señales.
- int [aleat_num](#) (int inf, int sup)
Devuelve un numero aleatorio entre inf y sup En caso de pasar un numero negativo se cambia de signo En caso de que sup sea menor inf se permutan.
- int [BubbleSort](#) (int *ganadores, double *pago, int ip, int iu)
Funcion que ordena la tabla ganadores en funcion de la tabla pago.

4.2.1 Detailed Description

Libreria para las funciones de la carrera entre las que se encuentran las funciones de cada proceso hijo y padre, hilos y auxiliares.

Author

Lucia Rivas Molina
Daniel Santo-Tomas Lopez

Date

9/05/2018

4.2.2 Function Documentation

4.2.2.1 `int aleat_num (int inf, int sup)`

Devuelve un numero aleatorio entre inf y sup En caso de pasar un numero negativo se cambia de signo En caso de que sup sea menor inf se permutan.

Parameters

<i>inf</i>	minimo numero aleatorio que puede salir
<i>sup</i>	maximo numero aleatorio que puede salir

Returns

el numero aleatorio

4.2.2.2 `int apostador (int dinero, int caballos, int apostadores, int id_cola)`

Funcion encargada de crear apostadores y mandar mensajes a las ventanillas.

Parameters

<i>dinero</i>	: dinero maximo por apostador introducido por teclado
<i>caballos</i>	: numero de caballos
<i>apostadores</i>	: numero de apostadores introducido por teclado
<i>id_cola</i>	: id de la cola de mensajes

Returns

OK o ERROR

4.2.2.3 int BubbleSort (int * *ganadores*, double * *pago*, int *ip*, int *iu*)

Funcion que ordena la tabla ganadores en funcion de la tabla pago.

Se utiliza para los 10 apostadores con mas ganancias

Parameters

<i>ganadores</i>	: tabla de los apostadores en orden de llegada
<i>pago</i>	: tabla del dinero de los apostadores en orden de llegada
<i>ip</i>	: primera iteracion
<i>iu</i>	: ultima iteracion

Returns

numero de iteraciones y ERROR en caso de error

4.2.2.4 int caballo (int *num*, int *longitud*, int * *pipelda*, int * *pipeVuelta*)

Funcion que realiza el trabajo de los caballos mandando y recibiendo posiciones y recorridos con el padre (principal)

Parameters

<i>num</i>	: numero de caballo que es
<i>pipelda</i>	: tuberias de escritura con los caballos
<i>pipeVuelta</i>	: tuberias de lectura con los caballos
<i>longitud</i>	: longitud de la carrera

Returns

OK o ERROR

4.2.2.5 void captura (int *sennal*)

Funcion capturadora de las señales.

Parameters

	SIGUSR1 para avisar al padre de que comienza la carrera SIGUSR2 para avisar al gestor y al apostador de que acaben SIGINT para acabar el programa con CTRL+C y borrar memoria
--	---

4.2.2.6 int gestor (sigset_t *mask*, int *id_cola*, int *caballos*, int *apostadores*, int *ventanillas*, int *id*)

Funcion encargada de inicializar las variables de la memoria compartida relacionadas con el dinero y de crear las ventanillas.

Parameters

<i>mask</i>	: mascara de señales permitidas
<i>id_cola</i>	: id de la cola de mensajes
<i>caballos</i>	: numero de caballos
<i>apostadores</i>	: numero de apostadores introducido por teclado
<i>ventanillas</i>	: numero de ventanillas introducido por teclado
<i>id</i>	: id de la cola de mensajes

Returns

OK o ERROR

4.2.2.7 `int monitor (sigset_t mask, int caballos, int longitud, int semid, Carrera_info * info, int dinero)`

Funcion encargada de imprimir por pantalla la informacion.

Parameters

<i>mask</i>	: mascara de señales permitidas
<i>caballos</i>	: numero de caballos
<i>longitud</i>	: longitud de la carrera
<i>semid</i>	: id de los semaforos
<i>info</i>	: estructura con la informacion de la carrera
<i>dinero</i>	: dinero maximo por apostador introducido por teclado

Returns

OK o ERROR

4.2.2.8 `int principal (sigset_t mask, int ** pipelda, int ** pipeVuelta, int caballos, int longitud, int semid, Carrera_info * info, pid_t * childpid)`

Funcion del proceso padre del programa, encargado de realizar la comunicacion entre procesos y de asignarle tiradas a los caballos.

Parameters

<i>mask</i>	: mascara de señales permitidas
<i>pipelda</i>	: tuberias de escritura con los caballos
<i>pipeVuelta</i>	: tuberias de lectura con los caballos
<i>caballos</i>	: numero de caballos
<i>longitud</i>	: longitud de la carrera
<i>semid</i>	: id de los semaforos
<i>info</i>	: estructura con la informacion de la carrera
<i>childpid</i>	: id de los procesos hijos

Returns

OK o ERROR

4.2.2.9 void* ventanilla (Hilo_args * arg)

Funcion de los hilos encargada de recibir las apuestas y agregarlas a la memoria compartida.

Parameters

<i>arg</i>	: estructura que contiene el id de la cola y el numero de ventanilla
------------	--

Returns

pthread_exit

4.3 lib.h File Reference

Libreria para las funciones de la carrera entre las que se encuentran las funciones de cada proceso hijo y padre, hilos y auxiliares.

```
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/msg.h>
#include <sys/wait.h>
#include <sys/shm.h>
#include <pthread.h>
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <errno.h>
#include <signal.h>
#include <time.h>
#include "semaforos.h"
```

Include dependency graph for lib.h: This graph shows which files directly or indirectly include this file:

Data Structures

- struct [Hilo_args](#)
Estructura pasada como argumento a los hilos num_ventana : numero de ventanilla que es id_mem : id de la memoria compartida.
- struct [Msg_apuesta](#)
Mensaje de cada apostador al gestor de apuestas.
- struct [Carrera_info](#)
Memoria compartida entre principal, gestor y monitor.

Macros

- `#define _GNU_SOURCE`
- `#define LIB_H`
- `#define MAX_CABALLOS 10`
maximo numero de caballos
- `#define MAX_APOSTADORES 100`
maximo numero de apostadores
- `#define FILEKEY "/bin/cat"`
file para la clave
- `#define KEY 1300`
key para los semaforos y la memoria compartida

Functions

- `int principal (sigset_t mask, int **pipelda, int **pipeVuelta, int caballos, int longitud, int semid, Carrera_info *info, pid_t *childpid)`
Funcion del proceso padre del programa, encargado de realizar la comunicacion entre procesos y de asignarle tiradas a los caballos.
- `int caballo (int num, int longitud, int *pipelda, int *pipeVuelta)`
Funcion que realiza el trabajo de los caballos mandando y recibiendo posiciones y recorridos con el padre (principal)
- `int monitor (sigset_t mask, int caballos, int longitud, int semid, Carrera_info *info, int dinero)`
Funcion encargada de imprimir por pantalla la informacion.
- `int apostador (int dinero, int caballos, int apostadores, int id_cola)`
Funcion encargada de crear apostadores y mandar mensajes a las ventanillas.
- `int gestor (sigset_t mask, int id_cola, int caballos, int apostadores, int ventanillas, int id)`
Funcion encargada de inicializar las variables de la memoria compartida relacionadas con el dinero y de crear las ventanillas.
- `void * ventanilla (Hilo_args *arg)`
Funcion de los hilos encargada de recibir las apuestas y agregarlas a la memoria compartida.
- `void captura (int sennal)`
Funcion capturadora de las señales.
- `int aleat_num (int inf, int sup)`
Devuelve un numero aleatorio entre inf y sup En caso de pasar un numero negativo se cambia de signo En caso de que sup sea menor inf se permutan.
- `int BubbleSort (int *ganadores, double *pago, int ip, int iu)`
Funcion que ordena la tabla ganadores en funcion de la tabla pago.

4.3.1 Detailed Description

Libreria para las funciones de la carrera entre las que se encuentran las funciones de cada proceso hijo y padre, hilos y auxiliares.

Author

Lucia Rivas Molina
Daniel Santo-Tomas Lopez

Date

9/05/2018

4.3.2 Function Documentation

4.3.2.1 int aleat_num (int *inf*, int *sup*)

Devuelve un numero aleatorio entre inf y sup En caso de pasar un numero negativo se cambia de signo En caso de que sup sea menor inf se permutan.

Parameters

<i>inf</i>	minimo numero aleatorio que puede salir
<i>sup</i>	maximo numero aleatorio que puede salir

Returns

el numero aleatorio

4.3.2.2 int apostador (int *dinero*, int *caballos*, int *apostadores*, int *id_cola*)

Funcion encargada de crear apostadores y mandar mensajes a las ventanillas.

Parameters

<i>dinero</i>	: dinero maximo por apostador introducido por teclado
<i>caballos</i>	: numero de caballos
<i>apostadores</i>	: numero de apostadores introducido por teclado
<i>id_cola</i>	: id de la cola de mensajes

Returns

OK o ERROR

4.3.2.3 int BubbleSort (int * *ganadores*, double * *pago*, int *ip*, int *iu*)

Funcion que ordena la tabla ganadores en funcion de la tabla pago.

Se utiliza para los 10 apostadores con mas ganancias

Parameters

<i>ganadores</i>	: tabla de los apostadores en orden de llegada
<i>pago</i>	: tabla del dinero de los apostadores en orden de llegada
<i>ip</i>	: primera iteracion
<i>iu</i>	: ultima iteracion

Returns

numero de iteraciones y ERROR en caso de error

4.3.2.4 int caballo (int num, int longitud, int * pipelda, int * pipeVuelta)

Funcion que realiza el trabajo de los caballos mandando y recibiendo posiciones y recorridos con el padre (principal)

Parameters

<i>num</i>	: numero de caballo que es
<i>pipelda</i>	: tuberias de escritura con los caballos
<i>pipeVuelta</i>	: tuberias de lectura con los caballos
<i>longitud</i>	: longitud de la carrera

Returns

OK o ERROR

4.3.2.5 void captura (int sennal)

Funcion capturadora de las señales.

Parameters

	SIGUSR1 para avisar al padre de que comienza la carrera SIGUSR2 para avisar al gestor y al apostador de que acaben SIGINT para acabar el programa con CTRL+C y borrar memoria
--	---

4.3.2.6 int gestor (sigset_t mask, int id_cola, int caballos, int apostadores, int ventanillas, int id)

Funcion encargada de inicializar las variables de la memoria compartida relacionadas con el dinero y de crear las ventanillas.

Parameters

<i>mask</i>	: mascara de señales permitidas
<i>id_cola</i>	: id de la cola de mensajes
<i>caballos</i>	: numero de caballos
<i>apostadores</i>	: numero de apostadores introducido por teclado
<i>ventanillas</i>	: numero de ventanillas introducido por teclado
<i>id</i>	: id de la cola de mensajes

Returns

OK o ERROR

4.3.2.7 int monitor (sigset_t mask, int caballos, int longitud, int semid, Carrera_info * info, int dinero)

Funcion encargada de imprimir por pantalla la informacion.

Parameters

<i>mask</i>	: mascara de señales permitidas
<i>caballos</i>	: numero de caballos
<i>longitud</i>	: longitud de la carrera
<i>semid</i>	: id de los semaforos
<i>info</i>	: estructura con la informacion de la carrera
<i>dinero</i>	: dinero maximo por apostador introducido por teclado

Returns

OK o ERROR

4.3.2.8 int principal (sigset_t mask, int ** pipelda, int ** pipeVuelta, int caballos, int longitud, int semid, Carrera_info * info, pid_t * childpid)

Funcion del proceso padre del programa, encargado de realizar la comunicacion entre procesos y de asignarle tiradas a los caballos.

Parameters

<i>mask</i>	: mascara de señales permitidas
<i>pipelda</i>	: tuberias de escritura con los caballos
<i>pipeVuelta</i>	: tuberias de lectura con los caballos
<i>caballos</i>	: numero de caballos
<i>longitud</i>	: longitud de la carrera
<i>semid</i>	: id de los semaforos
<i>info</i>	: estructura con la informacion de la carrera
<i>childpid</i>	: id de los procesos hijos

Returns

OK o ERROR

4.3.2.9 void* ventanilla (Hilo_args * arg)

Funcion de los hilos encargada de recibir las apuestas y agregarlas a la memoria compartida.

Parameters

<i>arg</i>	: estructura que contiene el id de la cola y el numero de ventanilla
------------	--

Returns

pthread_exit

4.4 semaforos.c File Reference

Biblioteca de semaforos.

```
#include "semaforos.h"
```

Include dependency graph for semaforos.c:

Functions

- int [Inicializar_Semaforo](#) (int semid, unsigned short *array)
Inicializa los semaforos indicados.
- int [Borrar_Semaforo](#) (int semid)
Elimina los semaforos indicados.
- int [Crear_Semaforo](#) (key_t key, int size, int *semid)
Crea los semaforos indicados.
- int [Down_Semaforo](#) (int id, int num_sem, int undo)
Hace un down de un semaforo.
- int [DownMultiple_Semaforo](#) (int id, int size, int undo, int *active)
Hace un down de un grupo de semaforos.
- int [Up_Semaforo](#) (int id, int num_sem, int undo)
Hace un up de un semaforo.
- int [UpMultiple_Semaforo](#) (int id, int size, int undo, int *active)
Hace un up de un grupo de semaforos.

4.4.1 Detailed Description

Biblioteca de semaforos.

Biblioteca de semaforos para las prácticas de SOPER

Author

Lucia Rivas Molina lucia.rivasmolina@estudiante.uam.es

Daniel Santo-Tomas Lopez daniel.santo-tomas@estudiante.uam.es

Date

20/03/2018

4.4.2 Function Documentation

4.4.2.1 int Borrar_Semaforo (int semid)

Elimina los semaforos indicados.

Parameters

<i>semid</i>	: identificador del grupo de semaforos
--------------	--

Returns

OK / ERROR

4.4.2.2 int Crear_Semaforo (key_t key, int size, int * semid)

Crea los semaforos indicados.

Parameters

<i>key</i>	: clave precompartida del semaforo
<i>size</i>	: numero de semaforos a crear
<i>semid</i>	: identificador del grupo de semaforos

Returns

OK / ERROR

4.4.2.3 int Down_Semaforo (int id, int num_sem, int undo)

Hace un down de un semaforo.

Parameters

<i>id</i>	: identificador del grupo de semaforos
<i>num_sem</i>	: numero del semaforo del cual hacemos el down
<i>undo</i>	: flags

Returns

OK / ERROR

4.4.2.4 int DownMultiple_Semaforo (int id, int size, int undo, int * active)

Hace un down de un grupo de semaforos.

Parameters

<i>id</i>	: identificador del grupo de semaforos
<i>size</i>	: numero de semaforos para hacer un down
<i>undo</i>	: flags
<i>active</i>	: semaforos involucrados

Returns

OK / ERROR

4.4.2.5 int Inicializar_Semaforo (int *semid*, unsigned short * *array*)

Inicializa los semaforos indicados.

Parameters

<i>semid</i>	: identificador del grupo de semaforos
<i>array</i>	: valores iniciales del semaforo

Returns

OK / ERROR

4.4.2.6 int Up_Semaforo (int *id*, int *num_sem*, int *undo*)

Hace un up de un semaforo.

Parameters

<i>id</i>	: identificador del grupo de semaforos
<i>num_sem</i>	: numero del semaforo del cual hacemos el down
<i>undo</i>	: flags

Returns

OK / ERROR

4.4.2.7 int UpMultiple_Semaforo (int *id*, int *size*, int *undo*, int * *active*)

Hace un up de un grupo de semaforos.

Parameters

<i>id</i>	: identificador del grupo de semaforos
<i>size</i>	: numero de semaforos para hacer un down
<i>undo</i>	: flags
<i>active</i>	: semaforos involucrados

Returns

OK / ERROR

Index

- aleat_num
 - lib.c, [8](#)
 - lib.h, [13](#)
- apostador
 - lib.c, [8](#)
 - lib.h, [13](#)
- Borrar_Semaforo
 - semaforos.c, [16](#)
- BubbleSort
 - lib.c, [8](#)
 - lib.h, [13](#)
- caballo
 - lib.c, [9](#)
 - lib.h, [14](#)
- captura
 - lib.c, [9](#)
 - lib.h, [14](#)
- carrera.c, [7](#)
- Carrera_info, [5](#)
- Crear_Semaforo
 - semaforos.c, [17](#)
- Down_Semaforo
 - semaforos.c, [17](#)
- DownMultiple_Semaforo
 - semaforos.c, [17](#)
- gestor
 - lib.c, [9](#)
 - lib.h, [14](#)
- Hilo_args, [5](#)
- Inicializar_Semaforo
 - semaforos.c, [18](#)
- lib.c, [7](#)
 - aleat_num, [8](#)
 - apostador, [8](#)
 - BubbleSort, [8](#)
 - caballo, [9](#)
 - captura, [9](#)
 - gestor, [9](#)
 - monitor, [10](#)
 - principal, [10](#)
 - ventanilla, [11](#)
- lib.h, [11](#)
 - aleat_num, [13](#)
 - apostador, [13](#)
- BubbleSort, [13](#)
- caballo, [14](#)
- captura, [14](#)
- gestor, [14](#)
- monitor, [14](#)
- principal, [15](#)
- ventanilla, [15](#)
- monitor
 - lib.c, [10](#)
 - lib.h, [14](#)
- Msg_apuesta, [6](#)
- principal
 - lib.c, [10](#)
 - lib.h, [15](#)
- semaforos.c, [16](#)
 - Borrar_Semaforo, [16](#)
 - Crear_Semaforo, [17](#)
 - Down_Semaforo, [17](#)
 - DownMultiple_Semaforo, [17](#)
 - Inicializar_Semaforo, [18](#)
 - Up_Semaforo, [18](#)
 - UpMultiple_Semaforo, [18](#)
- Up_Semaforo
 - semaforos.c, [18](#)
- UpMultiple_Semaforo
 - semaforos.c, [18](#)
- ventanilla
 - lib.c, [11](#)
 - lib.h, [15](#)