

PRAC.3 - G2202 - P6

Generated by Doxygen 1.8.11

Contents

1	Class Index	1
1.1	Class List	1
2	File Index	3
2.1	File List	3
3	Class Documentation	5
3.1	escritura Struct Reference	5
3.1.1	Detailed Description	5
3.1.2	Member Data Documentation	5
3.1.2.1	nnum	5
3.1.2.2	pathname	5
3.2	info Struct Reference	6
3.2.1	Detailed Description	6
3.2.2	Member Data Documentation	6
3.2.2.1	id	6
3.2.2.2	nombre	6
3.3	message Struct Reference	6
3.3.1	Detailed Description	6
3.3.2	Member Data Documentation	7
3.3.2.1	mtype	7
3.3.2.2	texto	7
3.4	productos Struct Reference	7
3.4.1	Detailed Description	7
3.4.2	Member Data Documentation	7
3.4.2.1	p	7

4	File Documentation	9
4.1	ejercicio2.c File Reference	9
4.1.1	Detailed Description	10
4.1.2	Function Documentation	10
4.1.2.1	handle_SIGUSR1(int sig)	10
4.1.2.2	rutina_hijo()	10
4.1.3	Variable Documentation	10
4.1.3.1	buff	10
4.2	ejercicio2_solved.c File Reference	11
4.2.1	Detailed Description	11
4.2.2	Function Documentation	12
4.2.2.1	handle_SIGUSR1(int sig)	12
4.2.2.2	rutina_hijo()	12
4.2.3	Variable Documentation	12
4.2.3.1	buff	12
4.2.3.2	semshm	12
4.3	ejercicio3.c File Reference	12
4.3.1	Detailed Description	13
4.3.2	Macro Definition Documentation	14
4.3.2.1	KEY1	14
4.3.2.2	KEY2	14
4.3.2.3	KEY3	14
4.3.2.4	KEY4	14
4.3.2.5	LET	14
4.3.2.6	NUM	14
4.3.2.7	PATH	14
4.3.3	Typedef Documentation	14
4.3.3.1	Productos	14
4.3.4	Function Documentation	15
4.3.4.1	consumidor()	15

4.3.4.2	producer()	15
4.3.5	Variable Documentation	15
4.3.5.1	buff	15
4.3.5.2	lleno	15
4.3.5.3	semshm	15
4.3.5.4	vacio	15
4.4	ejercicio4.c File Reference	15
4.4.1	Detailed Description	16
4.4.2	Macro Definition Documentation	16
4.4.2.1	ESCRITOR	16
4.4.2.2	LECTOR	17
4.4.2.3	MAX_BUF	17
4.4.2.4	MAX_NUM	17
4.4.2.5	MAX_RAND	17
4.4.2.6	MIN_NUM	17
4.4.2.7	MIN_RAND	17
4.4.2.8	NUM_HILOS	17
4.4.3	Function Documentation	17
4.4.3.1	escribir(void *escr)	17
4.4.3.2	leer(void *fdesc)	17
4.4.3.3	usage()	18
4.5	ejercicio5.c File Reference	18
4.5.1	Detailed Description	19
4.5.2	Macro Definition Documentation	19
4.5.2.1	KEY	19
4.5.2.2	MAX_MEM	19
4.5.2.3	NUM_PROC	19
4.5.2.4	PATH	19
4.5.3	Function Documentation	19
4.5.3.1	a(char *f_in)	19

4.5.3.2	b(pid_t pid_aux)	20
4.5.3.3	c(char *f_out, pid_t pid_aux)	20
4.5.3.4	shift_letras(char *mensaje)	20
4.5.3.5	usage()	20
4.6	mylib.c File Reference	20
4.6.1	Detailed Description	21
4.6.2	Function Documentation	21
4.6.2.1	aredigits(const char *string)	21
4.6.2.2	randNum(float inf, float sup)	21
4.6.2.3	sigaddset_var(sigset_t *sig, int sig,...)	22
4.7	mylib.h File Reference	22
4.7.1	Detailed Description	23
4.7.2	Function Documentation	23
4.7.2.1	aredigits(const char *string)	23
4.7.2.2	randNum(float inf, float sup)	23
4.7.2.3	sigaddset_var(sigset_t *sig, int sig,...)	24
4.8	semaforos.c File Reference	24
4.8.1	Detailed Description	25
4.8.2	Function Documentation	25
4.8.2.1	borrar_semaforo(int semid)	25
4.8.2.2	crear_semaforo(key_t key, int size, int *semid)	25
4.8.2.3	down_multiple_semaforo(int semid, int size, int undo, int *active)	26
4.8.2.4	down_semaforo(int semid, int num_sem, int undo)	26
4.8.2.5	inicializar_semaforo(int semid, unsigned short *array)	26
4.8.2.6	up_multiple_semaforo(int semid, int size, int undo, int *active)	27
4.8.2.7	up_semaforo(int semid, int num_sem, int undo)	27
4.9	semaforos.h File Reference	27
4.9.1	Detailed Description	28
4.9.2	Macro Definition Documentation	28
4.9.2.1	ERROR	28
4.9.2.2	OK	28
4.9.3	Function Documentation	28
4.9.3.1	borrar_semaforo(int semid)	28
4.9.3.2	crear_semaforo(key_t key, int size, int *semid)	29
4.9.3.3	down_multiple_semaforo(int semid, int size, int undo, int *active)	29
4.9.3.4	down_semaforo(int semid, int num_sem, int undo)	29
4.9.3.5	inicializar_semaforo(int semid, unsigned short *array)	30
4.9.3.6	up_multiple_semaforo(int semid, int size, int undo, int *active)	30
4.9.3.7	up_semaforo(int semid, int num_sem, int undo)	30

Chapter 1

Class Index

1.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

escritura	Estructura para pasar información a los threads	5
info	Informacion del usuario	6
message	Estructura personalizada de mensaje	6
productos	Array de productos	7

Chapter 2

File Index

2.1 File List

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

ejercicio2.c	
Ejercicio 2	9
ejercicio2_lib.h	??
ejercicio2_solved.c	
Ejercicio 2 Solved	11
ejercicio3.c	
Ejercicio 3	12
ejercicio4.c	
Ejercicio 4	15
ejercicio5.c	
Ejercicio 5	18
mylib.c	
Funciones personales varias. Implementacion de mylib.h	20
mylib.h	
Interfaz de funciones personales varias	22
semaforos.c	
Funciones personales acerca de semaforos. Implementacion	24
semaforos.h	
Interfaz de funciones personales acercas de semaforos	27

Chapter 3

Class Documentation

3.1 escritura Struct Reference

Estructura para pasar información a los threads.

Public Attributes

- int [nnum](#)
- char [pathname](#) [[MAX_BUF](#)]

3.1.1 Detailed Description

Estructura para pasar información a los threads.

Contiene la información que necesitamos transmitir del padre al thread.

3.1.2 Member Data Documentation

3.1.2.1 int escritura::nnum

Número de números a generar

3.1.2.2 char escritura::pathname[MAX_BUF]

Pathname del fichero donde escribir los números

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

- [ejercicio4.c](#)

3.2 info Struct Reference

Informacion del usuario.

```
#include <ejercicio2_lib.h>
```

Public Attributes

- char [nombre](#) [MAX_NOMBRE]
- int [id](#)

3.2.1 Detailed Description

Informacion del usuario.

Esta estructura guarda la información de un usuario.

3.2.2 Member Data Documentation

3.2.2.1 int info::id

Id del usuario

3.2.2.2 char info::nombre[MAX_NOMBRE]

Nombre del usuario

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

- [ejercicio2_lib.h](#)

3.3 message Struct Reference

Estructura personalizada de mensaje.

Public Attributes

- long [mtype](#)
- char [texto](#) [MAX_MEM/sizeof(char)]

3.3.1 Detailed Description

Estructura personalizada de mensaje.

Esta estructura guarda el tipo de mensaje y un array de chars con la información.

3.3.2 Member Data Documentation

3.3.2.1 long message::mtype

Tipo de mensaje

3.3.2.2 char message::texto[MAX_MEM/sizeof(char)]

Información del mensaje

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

- [ejercicio5.c](#)

3.4 productos Struct Reference

Array de productos.

Public Attributes

- char [p](#) [[LET+NUM](#)]

3.4.1 Detailed Description

Array de productos.

Array de chars en los que se guardarán y leeran los productos

3.4.2 Member Data Documentation

3.4.2.1 char productos::p[LET+NUM]

Array de productos

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

- [ejercicio3.c](#)

Chapter 4

File Documentation

4.1 ejercicio2.c File Reference

Ejercicio 2.

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/ipc.h>
#include <sys/wait.h>
#include <string.h>
#include <errno.h>
#include <sys/shm.h>
#include <signal.h>
#include <stdbool.h>
#include "mylib.h"
#include "ejercicio2_lib.h"
```

Include dependency graph for ejercicio2.c:



Functions

- void **handle_SIGUSR1** (int sig)
Manejador de SIGUSR1.
- void **rutina_hijo** ()
Rutina que sigue el proceso hijo.
- int **main** (int argc, char const *argv[])

Variables

- Info * **buff** = NULL

4.1.1 Detailed Description

Ejercicio 2.

Este fichero contiene el código fuente del ejercicio 2 de la entrega.

Author

Rafael Sánchez & Sergio Galán

Version

1.0

Date

14-04-2018

4.1.2 Function Documentation

4.1.2.1 void handle_SIGUSR1 (int *sig*)

Manejador de SIGUSR1.

Imprime el nombre y el id del usuario en la memoria compartida a la recepcion de SIGUSR1.

4.1.2.2 void rutina_hijo ()

Rutina que sigue el proceso hijo.

Duerme al comienzo, lee de teclado y envía SIGUSR1 al padre.

4.1.3 Variable Documentation

4.1.3.1 Info* buff = NULL

Variable global donde se guarda la informacion de usuario

4.2 ejercicio2_solved.c File Reference

Ejercicio 2 Solved.

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/ipc.h>
#include <sys/wait.h>
#include <string.h>
#include <errno.h>
#include <sys/shm.h>
#include <signal.h>
#include <sys/types.h>
#include <stdbool.h>
#include <sys/sem.h>
#include <ctype.h>
#include "mylib.h"
#include "semaforos.h"
#include "ejercicio2_lib.h"
```

Include dependency graph for ejercicio2_solved.c:



Functions

- void `handle_SIGUSR1` (int sig)
Manejador de SIGUSR1.
- void `rutina_hijo` ()
Rutina que sigue el proceso hijo.
- int `main` (int argc, char const *argv[])

Variables

- Info * `buff` = NULL
- int `semshm`

4.2.1 Detailed Description

Ejercicio 2 Solved.

Este fichero contiene el código fuente del ejercicio 2 de la entrega.

Author

Rafael Sánchez & Sergio Galán

Version

1.0

Date

14-04-2018

4.2.2 Function Documentation

4.2.2.1 void handle_SIGUSR1 (int sig)

Manejador de SIGUSR1.

Imprime el nombre y el id del usuario en la memoria compartida a la recepcion de SIGUSR1. Levanta el semaforo.

4.2.2.2 void rutina_hijo ()

Rutina que sigue el proceso hijo.

Duerme al comienzo, reserva el recurso del input y la memoria, lee de teclado y envía SIGUSR1 al padre.

4.2.3 Variable Documentation

4.2.3.1 Info* buff = NULL

Variable global donde se guarda la informacion de usuario

4.2.3.2 int semshm

Semaforo para proteger el recurso

4.3 ejercicio3.c File Reference

Ejercicio 3.

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/ipc.h>
#include <sys/wait.h>
#include <string.h>
#include <errno.h>
#include <sys/shm.h>
#include <signal.h>
#include <sys/types.h>
#include <stdbool.h>
#include <sys/sem.h>
#include <ctype.h>
#include "mylib.h"
#include "semaforos.h"
Include dependency graph for ejercicio3.c:
```



Classes

- struct `productos`
Array de productos.

Macros

- #define `LET` 26
- #define `NUM` 10
- #define `KEY1` 1300
- #define `KEY2` 1400
- #define `KEY3` 1500
- #define `KEY4` 1600
- #define `PATH` "/bin/bash"

Typedefs

- typedef struct `productos` `Productos`
Array de productos.

Functions

- void `productor` ()
Rutina que sigue el proceso indicado como productor.
- void `consumidor` ()
Rutina que sigue el proceso indicado como consumidor.
- int `main` (int argc, char const *argv[])

Variables

- `Productos` * `buff` = NULL
- int `semshm`
- int `vacio`
- int `lleno`

4.3.1 Detailed Description

Ejercicio 3.

Este fichero contiene el código fuente del ejercicio 5 de la entrega.

Author

Rafael Sánchez & Sergio Galán

Version

1.0

Date

14-04-2018

4.3.2 Macro Definition Documentation

4.3.2.1 `#define KEY1 1300`

Numero para generar una key con ftok

4.3.2.2 `#define KEY2 1400`

Numero para generar una key con ftok

4.3.2.3 `#define KEY3 1500`

Numero para generar una key con ftok

4.3.2.4 `#define KEY4 1600`

Numero para generar una key con ftok

4.3.2.5 `#define LET 26`

Numero de letras en ascii

4.3.2.6 `#define NUM 10`

Numero de números en ascii

4.3.2.7 `#define PATH "/bin/bash"`

Path para generar una key con ftok

4.3.3 Typedef Documentation

4.3.3.1 `typedef struct productos Productos`

Array de productos.

Array de chars en los que se guardarán y leeran los productos

4.3.4 Function Documentation

4.3.4.1 void consumidor ()

Rutina que sigue el proceso indicado como consumidor.

Lee e imprime los caracteres escritos por el productor en la memoria compartida

4.3.4.2 void productor ()

Rutina que sigue el proceso indicado como productor.

Genera todas las letras y números del ascii en la memoria compartida

4.3.5 Variable Documentation

4.3.5.1 Productos* buff = NULL

Variable global donde se guardan los productos

4.3.5.2 int lleno

< Controla que el consumidor no consuma si el sitio esta vacio

4.3.5.3 int semshm

< Controla el acceso a la memoria compartida

4.3.5.4 int vacio

< Controla que el productor no produzca si el sitio esta lleno

4.4 ejercicio4.c File Reference

Ejercicio 4.

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <errno.h>
#include <sys/types.h>
#include <stdbool.h>
#include <pthread.h>
#include <sys/stat.h>
#include <sys/mman.h>
#include <fcntl.h>
#include "mylib.h"
#include "semaforos.h"
```

Include dependency graph for ejercicio4.c:



Classes

- struct [escritura](#)

Estructura para pasar información a los threads.

Macros

- #define [NUM_HILOS](#) 2
- #define [LECTOR](#) 0
- #define [ESCRITOR](#) 1
- #define [MIN_NUM](#) 1000
- #define [MAX_NUM](#) 2000
- #define [MIN_RAND](#) 100
- #define [MAX_RAND](#) 1000
- #define [MAX_BUF](#) 256

Functions

- void * [escribir](#) (void *escr)
Función que nos permite escribir en un fichero los números aleatorios requeridos.
- void * [leer](#) (void *fdesc)
Función que mapea un fichero y lee su información.
- void [usage](#) ()
Imprime el uso correcto del ejercicio 4.
- int **main** (int argc, char const *argv[])

4.4.1 Detailed Description

Ejercicio 4.

Este fichero contiene el código fuente del ejercicio 4 de la entrega.

Author

Rafael Sánchez & Sergio Galán

Version

1.0

Date

14-04-2018

4.4.2 Macro Definition Documentation

4.4.2.1 #define [ESCRITOR](#) 1

Numero que representa al escritor

4.4.2.2 `#define LECTOR 0`

Numero que representa al lector

4.4.2.3 `#define MAX_BUF 256`

Tamaño máximo del buffer

4.4.2.4 `#define MAX_NUM 2000`

Máximo número de números a generar

4.4.2.5 `#define MAX_RAND 1000`

Máximo número aleatorio

4.4.2.6 `#define MIN_NUM 1000`

Mínimo número de números a generar

4.4.2.7 `#define MIN_RAND 100`

Mínimo número aleatorio

4.4.2.8 `#define NUM_HILOS 2`

Numero de hilos

4.4.3 Function Documentation

4.4.3.1 `void * escribir (void * escri)`

Función que nos permite escribir en un fichero los números aleatorios requeridos.

Escribe en el fichero indicado por pathname nnum números aleatorios

4.4.3.2 `void * leer (void * fdesc)`

Función que mapea un fichero y lee su información.

Mapea el fichero indicado por fdesc a memoria y lee su información desde ahí

4.4.3.3 void usage ()

Imprime el uso correcto del ejercicio 4.

Imprime el uso correcto del ejercicio 2.

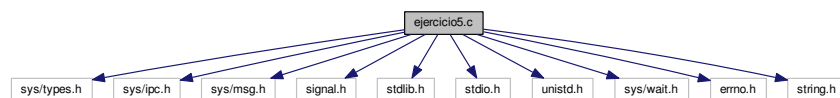
Imprime el orden de los parametros de entrada del ejercicio 4.

4.5 ejercicio5.c File Reference

Ejercicio 5.

```
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/msg.h>
#include <signal.h>
#include <stdlib.h>
#include <stdio.h>
#include <unistd.h>
#include <sys/wait.h>
#include <errno.h>
#include <string.h>
```

Include dependency graph for ejercicio5.c:



Classes

- struct [message](#)
Estructura personalizada de mensaje.

Macros

- #define [NUM_PROC](#) 3
- #define [MAX_MEM](#) 2048
- #define [KEY](#) 1300
- #define [PATH](#) "/bin/bash"

Functions

- void [a](#) (char *f_in)
Realiza la función del proceso A descrito en la práctica.
- void [b](#) (pid_t pid_aux)
Realiza la función del proceso B descrito en la práctica.
- void [c](#) (char *f_out, pid_t pid_aux)
Realiza la función del proceso C descrito en la práctica.
- void [usage](#) ()
Imprime el uso correcto del ejercicio 5.
- void [shift_letras](#) (char *mensaje)
Codifica los mensajes según lo descrito en la práctica.
- int [main](#) (int argc, char *argv[])

4.5.1 Detailed Description

Ejercicio 5.

Este fichero contiene el código fuente del ejercicio 5 de la entrega.

Author

Rafael Sánchez & Sergio Galán

Version

1.0

Date

14-04-2018

4.5.2 Macro Definition Documentation

4.5.2.1 `#define KEY 1300`

Número para generar una key con ftok

4.5.2.2 `#define MAX_MEM 2048`

Tamaño máximo de la información del mensaje

4.5.2.3 `#define NUM_PROC 3`

Número de procesos

4.5.2.4 `#define PATH "/bin/bash"`

Path para generar una key con ftok

4.5.3 Function Documentation

4.5.3.1 `void a (char * f_in)`

Realiza la función del proceso A descrito en la práctica.

Trocea el fichero en bloques de 2 kB (Salvo el último, que puede no serlo) y los manda como mensajes.

4.5.3.2 void b (pid_t pid_aux)

Realiza la función del proceso B descrito en la práctica.

Recibe los mensajes de A, los codifica con `shift_letras` y los manda como mensajes otra vez.

4.5.3.3 void c (char * f_out, pid_t pid_aux)

Realiza la función del proceso C descrito en la práctica.

Recibe los mensajes de B y los guarda en un fichero de texto.

4.5.3.4 void shift_letras (char * mensaje)

Codifica los mensajes según lo descrito en la práctica.

Hace un Cifrado César con clave 1 al mensaje recibido.

4.5.3.5 void usage ()

Imprime el uso correcto del ejercicio 5.

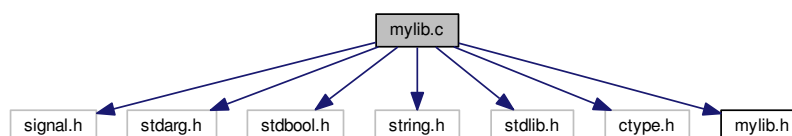
Imprime el uso correcto del ejercicio 2.

Imprime el orden de los parametros de entrada del ejercicio 5.

4.6 mylib.c File Reference

Funciones personales varias. Implementacion de [mylib.h](#).

```
#include <signal.h>
#include <stdarg.h>
#include <stdbool.h>
#include <string.h>
#include <stdlib.h>
#include <ctype.h>
#include "mylib.h"
Include dependency graph for mylib.c:
```



Functions

- void [sigaddset_var](#) (sigset_t *sig, int sig,...)
Añade una lista de señales a un set.
- float [randNum](#) (float inf, float sup)
Genera un número real aleatorio en el rango [inf, sup)
- bool [aredigits](#) (const char *string)
Comprueba si una string es numerica.

4.6.1 Detailed Description

Funciones personales varias. Implementacion de [mylib.h](#).

Este fichero contiene el código fuente de las funciones en la interfaz [mylib.h](#).

Author

Rafael Sánchez & Sergio Galán

Version

1.0

Date

12-04-2018

4.6.2 Function Documentation

4.6.2.1 bool aredigits (const char * *string*)

Comprueba si una string es numerica.

Comprueba si cada caracter de la cadena string es un valor numerico.

Parameters

<i>string</i>	Cadena de caracteres.
---------------	-----------------------

Returns

Devuelve true si la cadena es numerica, false si no lo es.

4.6.2.2 float randNum (float *inf*, float *sup*)

Genera un número real aleatorio en el rango [inf, sup)

Genera un número real aleatorio en el rango [inf, sup)

Parameters

<i>inf</i>	Límite inferior del intervalo
<i>sup</i>	Límite superior del intervalo

Returns

Devuelve el número aleatorio generado

4.6.2.3 void sigaddset_var (sigset_t * sigt, int sig, ...)

Añade una lista de señales a un set.

Añade una lista de señales acabada en -1 al set dado por sigt. Ejemplo de uso : sigaddset_var(sigt, SIGUSR1, SIGUSR2, SIGINT, -1);

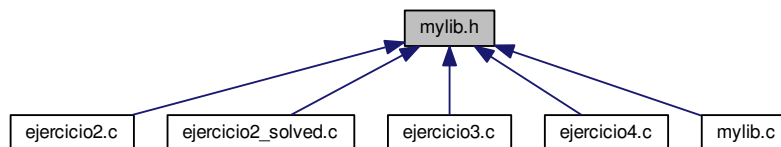
Parameters

<i>sigt</i>	Puntero de tipo sigset_t. Set de señales.
<i>sig</i>	Indices de las distintas señales. De extensión variable. Debe terminar en -1.

4.7 mylib.h File Reference

Interfaz de funciones personales varias.

This graph shows which files directly or indirectly include this file:

**Functions**

- void [sigaddset_var](#) (sigset_t *sigt, int sig,...)
Añade una lista de señales a un set.
- float [randNum](#) (float inf, float sup)
Genera un número real aleatorio en el rango [inf, sup)
- bool [aredigits](#) (const char *string)
Comprueba si una string es numerica.

4.7.1 Detailed Description

Interfaz de funciones personales varias.

Este fichero contiene las funciones personales varias.

Author

Rafael Sánchez & Sergio Galán

Version

1.0

Date

12-04-2018

4.7.2 Function Documentation

4.7.2.1 bool aredigits (const char * *string*)

Comprueba si una string es numerica.

Comprueba si cada caracter de la cadena string es un valor numerico.

Parameters

<i>string</i>	Cadena de caracteres.
---------------	-----------------------

Returns

Devuelve true si la cadena es numerica, false si no lo es.

4.7.2.2 float randNum (float *inf*, float *sup*)

Genera un número real aleatorio en el rango [inf, sup)

Genera un número real aleatorio en el rango [inf, sup)

Parameters

<i>inf</i>	Límite inferior del intervalo
<i>sup</i>	Límite superior del intervalo

Returns

Devuelve el número aleatorio generado

4.7.2.3 void sigaddset_var (sigset_t * sigt, int sig, ...)

Añade una lista de señales a un set.

Añade una lista de señales acabada en -1 al set dado por sigt. Ejemplo de uso : sigaddset_var(sigt, SIGUSR1, SIGUSR2, SIGINT, -1);

Parameters

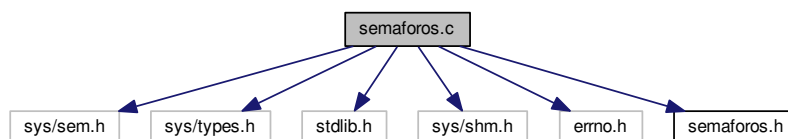
<i>sigt</i>	Puntero de tipo sigset_t. Set de señales.
<i>sig</i>	Indices de las distintas señales. De extensión variable. Debe terminar en -1.

4.8 semaforos.c File Reference

Funciones personales acerca de semaforos. Implementación.

```
#include <sys/sem.h>
#include <sys/types.h>
#include <stdlib.h>
#include <sys/shm.h>
#include <errno.h>
#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)
Borra un semáforo.
- int [crear_semaforo](#) (key_t key, int size, int *semid)
Crea un semaforo con la clave y el tamaño especificado. Lo inicializa a 0.
- int [down_semaforo](#) (int semid, int num_sem, int undo)
Baja el semaforo indicado.
- int [down_multiple_semaforo](#) (int semid, int size, int undo, int *active)

Baja todos los semaforos del array indicado por active.

- int [up_semaforo](#) (int semid, int num_sem, int undo)

Sube el semaforo indicado.

- int [up_multiple_semaforo](#) (int semid, int size, int undo, int *active)

Sube todos los semaforos del array indicado por active.

4.8.1 Detailed Description

Funciones personales acerca de semaforos. Implementacion.

Este fichero contiene el código fuente de las funciones en la interfaz [semaforos.h](#).

Author

Rafael Sánchez & Sergio Galán

Version

1.0

Date

06-04-2018

4.8.2 Function Documentation

4.8.2.1 int borrar_semaforo (int *semid*)

Borra un semáforo.

Parameters

<i>semid</i>	Identificador del semaforo.
--------------	-----------------------------

Returns

OK si todo fue correcto, ERROR en caso de error.

4.8.2.2 int crear_semaforo (key_t *key*, int *size*, int * *semid*)

Crea un semaforo con la clave y el tamaño especificado. Lo inicializa a 0.

Parameters

<i>key</i>	Clave precompartida del semaforo.
<i>size</i>	Tamaño del semaforo.
<i>semid</i>	Identificador del semaforo.

Returns

ERROR en caso de error, 0 si ha creado el semaforo, 1 si ya estaba creado.

4.8.2.3 int down_multiple_semaforo (int *semid*, int *size*, int *undo*, int * *active*)

Baja todos los semaforos del array indicado por active.

Parameters

<i>semid</i>	Identificador del semaforo.
<i>size</i>	Tamaño de active.
<i>undo</i>	Flag de modo persistente pese a finalización abrupta.
<i>active</i>	Semaforos involucrados. Array de enteros.

Returns

int: OK si todo fue correcto, ERROR en caso de error.

4.8.2.4 int down_semaforo (int *semid*, int *num_sem*, int *undo*)

Baja el semaforo indicado.

Parameters

<i>semid</i>	Identificador del semaforo.
<i>num_sem</i>	Semaforo dentro del array.
<i>undo</i>	Flag de modo persistente pese a finalización abrupta.

Returns

OK si todo fue correcto, ERROR en caso de error.

4.8.2.5 int inicializar_semaforo (int *semid*, unsigned short * *array*)

Inicializa los semaforos indicados.

Parameters

<i>semid</i>	Identificador del semaforo.
<i>array</i>	Valores iniciales.

Returns

OK si todo fue correcto, ERROR en caso de error.

4.8.2.6 int up_multiple_semaforo (int *semid*, int *size*, int *undo*, int * *active*)

Sube todos los semaforos del array indicado por active.

Parameters

<i>semid</i>	Identificador del semaforo.
<i>size</i>	Tamaño de active.
<i>undo</i>	Flag de modo persistente pese a finalización abrupta.
<i>active</i>	Semaforos involucrados. Array de enteros.

Returns

int: OK si todo fue correcto, ERROR en caso de error.

4.8.2.7 int up_semaforo (int *semid*, int *num_sem*, int *undo*)

Sube el semaforo indicado.

Parameters

<i>semid</i>	Identificador del semaforo.
<i>num_sem</i>	Semaforo dentro del array.
<i>undo</i>	Flag de modo persistente pese a finalización abrupta.

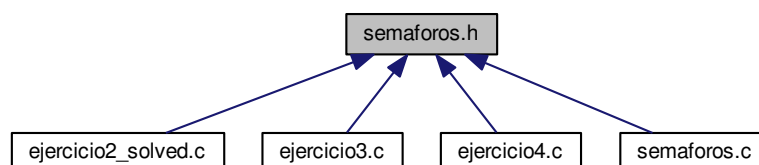
Returns

OK si todo fue correcto, ERROR en caso de error.

4.9 semaforos.h File Reference

Interfaz de funciones personales acerca de semaforos.

This graph shows which files directly or indirectly include this file:



Macros

- #define OK EXIT_SUCCESS
- #define ERROR -1

Functions

- int [inicializar_semaforo](#) (int semid, unsigned short *array)
Inicializa los semaforos indicados.
- int [borrar_semaforo](#) (int semid)
Borra un semáforo.
- int [crear_semaforo](#) (key_t key, int size, int *semid)
Crea un semaforo con la clave y el tamaño especificado. Lo inicializa a 0.
- int [down_semaforo](#) (int semid, int num_sem, int undo)
Baja el semaforo indicado.
- int [down_multiple_semaforo](#) (int semid, int size, int undo, int *active)
Baja todos los semaforos del array indicado por active.
- int [up_semaforo](#) (int semid, int num_sem, int undo)
Sube el semaforo indicado.
- int [up_multiple_semaforo](#) (int semid, int size, int undo, int *active)
Sube todos los semaforos del array indicado por active.

4.9.1 Detailed Description

Interfaz de funciones personales acerca de semaforos.

Este fichero contiene las funciones personales acerca de semaforos.

Author

Profesores de SOPER (EDIT: Rafael Sánchez & Sergio Galán).

Version

1.0

Date

06-04-2018

4.9.2 Macro Definition Documentation

4.9.2.1 #define ERROR -1

Error

4.9.2.2 #define OK EXIT_SUCCESS

Todo fue bien

4.9.3 Function Documentation

4.9.3.1 int borrar_semaforo (int semid)

Borra un semáforo.

Parameters

<i>semid</i>	Identificador del semaforo.
--------------	-----------------------------

Returns

OK si todo fue correcto, ERROR en caso de error.

4.9.3.2 `int crear_semaforo (key_t key, int size, int * semid)`

Crea un semaforo con la clave y el tamaño especificado. Lo inicializa a 0.

Parameters

<i>key</i>	Clave precompartida del semaforo.
<i>size</i>	Tamaño del semaforo.
<i>semid</i>	Identificador del semaforo.

Returns

ERROR en caso de error, 0 si ha creado el semaforo, 1 si ya estaba creado.

4.9.3.3 `int down_multiple_semaforo (int semid, int size, int undo, int * active)`

Baja todos los semaforos del array indicado por active.

Parameters

<i>semid</i>	Identificador del semaforo.
<i>size</i>	Tamaño de active.
<i>undo</i>	Flag de modo persistente pese a finalización abrupta.
<i>active</i>	Semaforos involucrados. Array de enteros.

Returns

int: OK si todo fue correcto, ERROR en caso de error.

4.9.3.4 `int down_semaforo (int semid, int num_sem, int undo)`

Baja el semaforo indicado.

Parameters

<i>semid</i>	Identificador del semaforo.
<i>num_sem</i>	Semaforo dentro del array.
<i>undo</i>	Flag de modo persistente pese a finalización abrupta.

Returns

OK si todo fue correcto, ERROR en caso de error.

4.9.3.5 int inicializar_semaforo (int *semid*, unsigned short * *array*)

Inicializa los semaforos indicados.

Parameters

<i>semid</i>	Identificador del semaforo.
<i>array</i>	Valores iniciales.

Returns

OK si todo fue correcto, ERROR en caso de error.

4.9.3.6 int up_multiple_semaforo (int *semid*, int *size*, int *undo*, int * *active*)

Sube todos los semaforos del array indicado por active.

Parameters

<i>semid</i>	Identificador del semaforo.
<i>size</i>	Tamaño de active.
<i>undo</i>	Flag de modo persistente pese a finalización abrupta.
<i>active</i>	Semaforos involucrados. Array de enteros.

Returns

int: OK si todo fue correcto, ERROR en caso de error.

4.9.3.7 int up_semaforo (int *semid*, int *num_sem*, int *undo*)

Sube el semaforo indicado.

Parameters

<i>semid</i>	Identificador del semaforo.
<i>num_sem</i>	Semaforo dentro del array.
<i>undo</i>	Flag de modo persistente pese a finalización abrupta.

Returns

OK si todo fue correcto, ERROR en caso de error.

Index

a

ejercicio5.c, 19

aredigits

mylib.c, 21

mylib.h, 23

b

ejercicio5.c, 19

borrar_semaforo

semaforos.c, 25

semaforos.h, 28

buff

ejercicio2.c, 10

ejercicio2_solved.c, 12

ejercicio3.c, 15

c

ejercicio5.c, 20

consumidor

ejercicio3.c, 15

crear_semaforo

semaforos.c, 25

semaforos.h, 29

down_multiple_semaforo

semaforos.c, 26

semaforos.h, 29

down_semaforo

semaforos.c, 26

semaforos.h, 29

ERROR

semaforos.h, 28

ESCRITOR

ejercicio4.c, 16

ejercicio2.c, 9

buff, 10

handle_SIGUSR1, 10

rutina_hijo, 10

ejercicio2_solved.c, 11

buff, 12

handle_SIGUSR1, 12

rutina_hijo, 12

semshm, 12

ejercicio3.c, 12

buff, 15

consumidor, 15

KEY1, 14

KEY2, 14

KEY3, 14

KEY4, 14

LET, 14

lleno, 15

NUM, 14

PATH, 14

productor, 15

Productos, 14

semshm, 15

vacio, 15

ejercicio4.c, 15

ESCRITOR, 16

escribir, 17

LECTOR, 16

leer, 17

MAX_BUF, 17

MAX_NUM, 17

MAX_RAND, 17

MIN_NUM, 17

MIN_RAND, 17

NUM_HILOS, 17

usage, 17

ejercicio5.c, 18

a, 19

b, 19

c, 20

KEY, 19

MAX_MEM, 19

NUM_PROC, 19

PATH, 19

shift_letras, 20

usage, 20

escribir

ejercicio4.c, 17

escritura, 5

nnum, 5

pathname, 5

handle_SIGUSR1

ejercicio2.c, 10

ejercicio2_solved.c, 12

id

info, 6

info, 6

id, 6

nombre, 6

inicializar_semaforo

semaforos.c, 26

semaforos.h, 30

KEY1
 ejercicio3.c, 14
 KEY2
 ejercicio3.c, 14
 KEY3
 ejercicio3.c, 14
 KEY4
 ejercicio3.c, 14
 KEY
 ejercicio5.c, 19

 LECTOR
 ejercicio4.c, 16
 LET
 ejercicio3.c, 14
 leer
 ejercicio4.c, 17
 lleno
 ejercicio3.c, 15

 MAX_BUF
 ejercicio4.c, 17
 MAX_MEM
 ejercicio5.c, 19
 MAX_NUM
 ejercicio4.c, 17
 MAX_RAND
 ejercicio4.c, 17
 MIN_NUM
 ejercicio4.c, 17
 MIN_RAND
 ejercicio4.c, 17
 message, 6
 mtype, 7
 texto, 7
 mtype
 message, 7
 mylib.c, 20
 aredigits, 21
 randNum, 21
 sigaddset_var, 22
 mylib.h, 22
 aredigits, 23
 randNum, 23
 sigaddset_var, 24

 NUM_HILOS
 ejercicio4.c, 17
 NUM_PROC
 ejercicio5.c, 19
 NUM
 ejercicio3.c, 14
 nnum
 escritura, 5
 nombre
 info, 6

 OK
 semaforos.h, 28

 p
 productos, 7
 PATH
 ejercicio3.c, 14
 ejercicio5.c, 19
 pathname
 escritura, 5
 productor
 ejercicio3.c, 15
 Productos
 ejercicio3.c, 14
 productos, 7
 p, 7

 randNum
 mylib.c, 21
 mylib.h, 23
 rutina_hijo
 ejercicio2.c, 10
 ejercicio2_solved.c, 12

 semaforos.c, 24
 borrar_semaforo, 25
 crear_semaforo, 25
 down_multiple_semaforo, 26
 down_semaforo, 26
 inicializar_semaforo, 26
 up_multiple_semaforo, 26
 up_semaforo, 27
 semaforos.h, 27
 borrar_semaforo, 28
 crear_semaforo, 29
 down_multiple_semaforo, 29
 down_semaforo, 29
 ERROR, 28
 inicializar_semaforo, 30
 OK, 28
 up_multiple_semaforo, 30
 up_semaforo, 30
 semshm
 ejercicio2_solved.c, 12
 ejercicio3.c, 15
 shift_letras
 ejercicio5.c, 20
 sigaddset_var
 mylib.c, 22
 mylib.h, 24

 texto
 message, 7

 up_multiple_semaforo
 semaforos.c, 26
 semaforos.h, 30
 up_semaforo
 semaforos.c, 27
 semaforos.h, 30
 usage
 ejercicio4.c, 17

ejercicio5.c, [20](#)

vacio

ejercicio3.c, [15](#)