PRAC.3 - G2202 - P6

Generated by Doxygen 1.8.11

Contents

Chapter 1

Class Index

1.1 Class List

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

Chapter 2

File Index

2.1 File List

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

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

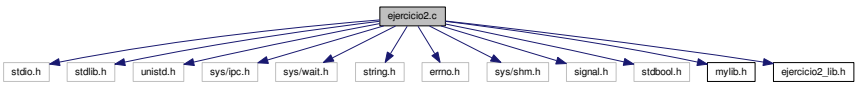
File Documentation

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

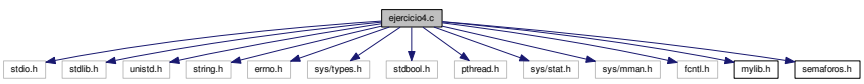
< Controla que el productor no produzca si el sitio esta lleno

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 * *escr*)

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.

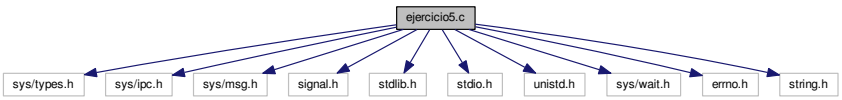
Imprime el orden de los parametros de entrada del ejercicio 4.

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.

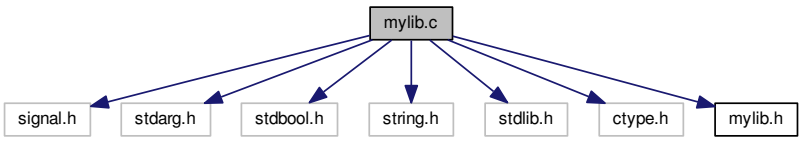
Imprime el orden de los parametros de entrada del ejercicio 5.

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 *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.

4.6.1 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

string	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

inf	Límite inferior del intervalo
sup	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);

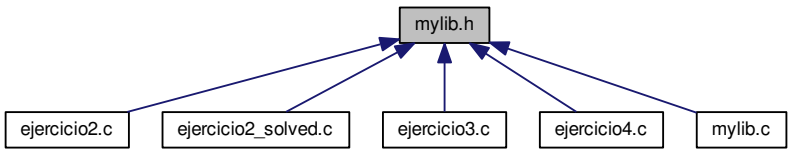
Parameters

sigt	Puntero de tipo sigset_t. Set de señales.
sig	Indices de las distinas señales. De extension 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

string	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

inf	Límite inferior del intervalo
sup	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

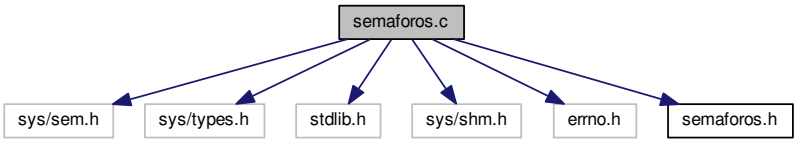
sigt	Puntero de tipo sigset_t. Set de señales.
sig	Indices de las distinas señales. De extension variable. Debe terminar en -1.

4.8 semaforos.c File Reference

Funciones personales acerca de semaforos. Implementacion.

```
#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

semid	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

key	Clave precompartida del semaforo.
size	Tamaño del semaforo.
semid	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

semid	Identificador del semaforo.
size	Tamaño de active.
undo	Flag de modo persistente pese a finalización abrupta.
active	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

semid	Identificador del semaforo.
num_sem	Semaforo dentro del array.
undo	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

semid	Identificador del semaforo.
array	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

semid	Identificador del semaforo.
size	Tamaño de active.
undo	Flag de modo persistente pese a finalización abrupta.
active	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

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

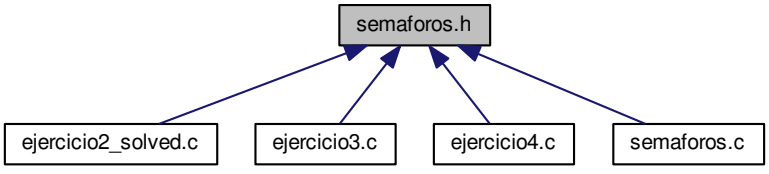
Returns

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

4.9 semaforos.h File Reference

Interfaz de funciones personales acercas 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 acercas 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

semid	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

key	Clave precompartida del semaforo.
size	Tamaño del semaforo.
semid	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

semid	Identificador del semaforo.
size	Tamaño de active.
undo	Flag de modo persistente pese a finalización abrupta.
active	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

semid	Identificador del semaforo.
num_sem	Semaforo dentro del array.
undo	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

semid	Identificador del semaforo.
array	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

semid	Identificador del semaforo.
size	Tamaño de active.
undo	Flag de modo persistente pese a finalización abrupta.
active	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

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

Returns

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

Index