My Project

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 _Info Struct Reference

Public Attributes

- char **nombre** [80]
- int **id**

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

- ejercicio2.c

3.2 AlphaStack Struct Reference

Public Attributes

- char **alpha** [27]
- int **end**
- int **temp**

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

- ejercicio6.c

Chapter 4

File Documentation

4.1 ejercicio2.c File Reference

fichero que contiene la implementacion del ejercicio 2

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

Include dependency graph for ejercicio2.c:

4.2 ejercicio6.c File Reference

fichero que contiene ej6 de la practica 3 de SOPER para el problema productor consumidor

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

Include dependency graph for ejercicio6.c:

Classes

- struct AlphaStack

Macros

- #define **KEY** 14327876
- #define **FILEKEY** "/"

Functions

- int main (int argc, char **argv)
- int crear_shmem (char *filekey, int key, int size, AlphaStack **att)
- int crear_sems (char *filekey, int *mutex, int *lleno, int *vacio)
- int productor (AlphaStack *alpha, int *mutex, int *lleno, int *vacio, int prodSleep)
- int consumidor (AlphaStack *alpha, int *mutex, int *lleno, int *vacio, int consSleep)

4.2.1 Detailed Description

fichero que contiene ej6 de la practica 3 de SOPER para el problema productor consumidor

Author

Lucia Asencio y Rodrigo de Pool

Date

16-3-2017

4.2.2 Function Documentation

4.2.2.1 int consumidor (AlphaStack * *alpha,* int * *mutex,* int * *lleno,* int * *vacio,* int *consSleep*)

Tras crear los semaforos necesarios, el proceso que ejecuta esta funcion consume, mientras haya productos disponibles, las letras del abecedario En caso de error, hara detach de la memoria pero no la borrara

Parameters

alpha	puntero a la estructura compartida
mutex,lleno,vacio	punteros a los semids

Returns

-1 en caso de error, else 0

4.2.2.2 int crear_sems (char * *filekey,* int * *mutex,* int * *lleno,* int * *vacio*)

Solicita los 3 sems necesarios para el problema productor/consumidor Devuelve el identificador del semaforo en *mutex, *lleno, *vacio Los semaforos son inicializados: mutex a 1, lleno a 0, vacio a 26

Parameters

filekey	parametro para generar la clave con ftok. La key cambia para cada semaforo
mutex,lleno,vacio	punteros a los ints donde se guarda los sems id

Returns

0 si semaforos creados, -1 en caso de error (libera los semaforos ya creados si error)

4.2.2.3 int crear_shmem (char * *filekey,* int *key,* int *size,* AlphaStack ** *att*)

Solicita zona de memoria compartida, primero con IPC_CREAT | IPC_EXCL y, si ya existe, de nuevo sin estas banderas. Also attaches the result to a struct AlphaQueue

Parameters

filekey,key	: parametros identicos a la funcion ftok
size	int > 0, tamanio de la memoria a compartir

Returns

identifier for the shared mmemory, -1 in case of error

4.2.2.4 int main (int *argc,* char ** *argv*)

Main que, utilizando dos procesos hijos, simula los procesos productor y consumidor Pueden omitirse todos los argumentos (los argumentos por defecto son 1000,5000, 80000), omitirse el ultimo (por defecto, se usa 100000), o no omitirse ninguno.

- argv[1] , n1, int >=0 que indica el usleep(n1) que ejecutara el padre entre produccion y produccion
- argv[2] , n2, int >=0 que indica el usleep(n2) que ejecutara el padre entre consumicion y consumicion
- argv[3] , n3, int > 0 que indica el usleep(n3) que ejecutara el hijo temporizador antes de cortar la produccion/consumicion de los otros dos.

  Returns

  0 si todo fue bien, -1 si error

4.2.2.5 int productor (AlphaStack * *alpha,* int * *mutex,* int * *lleno,* int * *vacio,* int *prodSleep*)

Tras crear los semaforos necesarios, esta funcion produce una cola de letras mientras no este disponible todo el abecedario En caso de error, hara detach de la memoria pero no la borrara

Parameters

alpha	puntero a la estructura compartida
mutex,lleno,vacio	punteros a los semids

Returns

-1 en caso de error, else 0.

4.3 main_test.c File Reference

fichero que contiene el main que prueba la libreria de semaforos

```
#include <stdio.h>
#include <stdlib.h>
#include "semaforostest.h"
```

Include dependency graph for main_test.c:

Functions

- void main ()

4.3.1 Detailed Description

fichero que contiene el main que prueba la libreria de semaforos

Author

Lucia Asencio y Rodrigo de Pool

Date

31-3-2017

4.3.2 Function Documentation

4.3.2.1 void main ( )

Este programa prueba todos los tests de la libreriía semaforostest.h y se asegura de que el retorno sea positivo

4.4 semaforos.c File Reference

fichero que contiene la implementacion de la libreria de semaforos

```
#include <sys/sem.h>
#include <sys/shm.h>
#include <stdio.h>
#include <stdlib.h>
#include "semaforos.h"
```

Include dependency graph for semaforos.c:

Functions

- int Crear_Semaforo (key_t key, int size, int *semid)
- int Inicializar_Semaforo (int semid, unsigned short *array)
- int Borrar_Semaforo (int semid)
- int operacionSemaforo (int id, int num_sem, int undo, int op)
- int Down_Semaforo (int id, int num_sem, int undo)
- int Up_Semaforo (int id, int num_sem, int undo)
- int operacionMultipleSemaforo (int id, int size, int undo, int *active, int op)
- int DownMultiple_Semaforo (int id, int size, int undo, int *active)
- int UpMultiple_Semaforo (int id, int size, int undo, int *active)

4.4.1 Detailed Description

fichero que contiene la implementacion de la libreria de semaforos

Author

Lucia Asencio y Rodrigo de Pool

Date

31-3-2017

4.4.2 Function Documentation

4.4.2.1 int Borrar_Semaforo (int *semid*)

Borra un array de semaforos

Parameters

semid	identificador del array

Returns

devuelve OK si todo correcto, y ERROR en caso de error

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

Crea un nuevo semaforo con el key dado y del tamanio indicado. Se inicializan los valores a 0 de todos los semaforos.

Parameters

key	clave para inicializar el semaforo
size	cantidad de semaforos que se quieren
semid	puntero a entero donde se guarda el identificador del array de semaforos

Returns

devuelve un 0 si el semaforo ya estaba creado 1 si se ha creado. Si se creo el semaforo en semid se guarda su Identificador Si ya estaba creado y no hubo error se devuelve el identificador, en este caso los valores del semaforo no son cambiados Si no se logra crear el semafoto devuelve ERROR en semid

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

Hace down a un semaforo

Parameters

id	identificador de array de semaforos
num_sem	numero de semaforo dentro del array
undo	flag que se quiere agregar (UNDO flag recomendable)

Returns

ERROR en caso de error, OK si todo correcto

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

Hace down sobre un conjunto de semaforos.

Parameters

id	identificador del array de semaforos
size	tamanio del array active
undo	flag a agregarle a la operacion
active	array con el conjunto de semaforos sobre el que se hace down

Returns

ERROR si hubo un error al hacer down en algun semaforo, OK en caso de que todo fue correctamente

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

Iniciliaza el semaforo a los valores indicados

Parameters

semid	id del semaforo
array	array de valores a asignar al semaforo

Returns

OK si no hubo errores, ERROR si los hubo

4.4.2.6 int operacionMultipleSemaforo (int *id,* int *size,* int *undo,* int * *active,* int *op*)

Funcion privada que generaliza el comportamiento de DownMultiple_Semaforo y UpMultiple_Semaforo. Se le pasa -1 si se quiere actuar como DownMultiple_Semaforo y 1 si se quiere actuar como UpMultiple_Semaforo

4.4.2.7 int operacionSemaforo (int *id,* int *num_sem,* int *undo,* int *op*)

Funcion privada que nos sirve para generalizar el codigo de down y up de un semaforo El unico cambio es que op es -1 y si es down y 1 si es up. Devuelve ERROR si hubo error o OK en caso de no haberlo

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

Hace up a un semaforo

Parameters

id	identificador de array de semaforos
num_sem	numero de semaforo dentro del array
undo	flag que se quiere agregar (UNDO flag recomendable)

Returns

ERROR en caso de error, OK si todo correcto

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

Hace up sobre un conjunto de semaforos.

Parameters

id	identificador del array de semaforos
size	tamanio del array active
undo	flag a agregarle a la operacion
active	array con el conjunto de semaforos sobre el que se hace down

Returns

ERROR si hubo un error al hacer up en algun semaforo, OK en caso de que todo fue correctamente

4.5 semaforostest.c File Reference

fichero que contiene la implementacion de la libreria de test de semaforos

```
#include "semaforostest.h"
#include "semaforos.h"
#include <stdio.h>
#include <stdlib.h>
#include <sys/sem.h>
```

Include dependency graph for semaforostest.c:

Functions

- int Crear_Semaforo_Test ()
- int Inicializar_Semaforo_Test ()
- int General_Single_Test (int num)
- int Down_Semaforo_Test ()
- int Up_Semaforo_Test ()
- int General_Multiple_Test (int num)
- int DownMultiple_Semaforo_Test ()
- int UpMultiple_Semaforo_Test ()

4.5.1 Detailed Description

fichero que contiene la implementacion de la libreria de test de semaforos

Author

Lucia Asencio y Rodrigo de Pool

Date

31-3-2017

4.5.2 Function Documentation

4.5.2.1 int Crear_Semaforo_Test ( )

Funcion que prueba Crear_Semaforo_Test: 1) Se asegura que se crear un semaforo en condiciones normales 2) Se asegura que se puede solicitar la informacion de un semaforo ya creado 3) Se asegura que NO se puede crear otro semaforo con la misma clave y un tamanio distinto (si fueran el mismo estariamos en el caso 2) 4) Se asegura que el valor de cada semaforo creado es 0

Returns

PASSED si el test termina con exito, NOT_PASSED en caso contrario

4.5.2.2 int Down_Semaforo_Test ( )

Funcion que prueba el correcto funcionamiento de Down_Semaforo: Crea un semaforo de 4 elementos y se inicializan. Luego se hace down a cada uno de los semaforos probando que, en efecto, su valor disminuye

Returns

PASSED si el test termina con exito, NOT_PASSED en caso contrario

4.5.2.3 int DownMultiple_Semaforo_Test ( )

Funcion que prueba el correcto funcioamiento de DownMultiple_Semaforo: Se crea un semaforo de 4 elementos y se inicializan. Se le aplica la funcion a 3 elementos indicando sus valores a traves de un array. Luego se prueba que los valores fueron cambiados satisfactoriamente y que el elemento no cambiado se mantuvo en su mismo estado

Returns

PASSED si el el test termina con exito, NOT_PASSED en caso contrario

4.5.2.4 int General_Multiple_Test (int *num*)

Funcion privada que generaliza las prueba DownMultiple_Semaforo_Test y UpMultiple_Semaforo_Test.

Parameters

num	si es -1 se realiza DownMultiple_Semaforo_Test, cualquier otro numero realiza UpMultiple_Semaforo_Test

Returns

PASSED o NOT_PASSED dependiendo de si se paso o no la prueba

4.5.2.5 int General_Single_Test (int *num*)

Funcion privada que generaliza las prueba Down_Semaforo_Test y Up_Semaforo_Test.

Parameters

num	si es -1 se realiza Down_Semaforo_Test, cualquier otro numero realiza Up_Semaforo_Test

Returns

PASSED o NOT_PASSED dependiendo de si se paso o no la prueba

4.5.2.6 int Inicializar_Semaforo_Test ( )

Funcion que se asegura del correcto funcionamiento de Inicializar_Semaforo: 1) Nos aseguramos de que podemos inicializar los valores correctamente 2) Nos aseguramos que devuelve error en caso de dar argumentos erroneos

Returns

PASSED si el test terminal con exito, en cualquier otro caso NOT_PASSED

4.5.2.7 int Up_Semaforo_Test ( )

Funcion que prueba el correcto funcionamiento de Up_Semaforo: Crea un semaforo de 4 elementos y se inicializan. Luego se hace up a cada uno de los semaforos probando que, en efecto, su valor aumenta

Returns

PASSED si el test termina con exito, NOT_PASSED en caso contrario

4.5.2.8 int UpMultiple_Semaforo_Test ( )

Funcion que prueba el correcto funcioamiento de UpMultiple_Semaforo: Se crea un semaforo de 4 elementos y se inicializan. Se le aplica la funcion a 3 elementos indicando sus valores a traves de un array. Luego se prueba que los valores fueron cambiados satisfactoriamente y que el elemento no cambiado se mantuvo en su mismo estado

Returns

PASSED si el el test termina con exito, NOT_PASSED en caso contrario

Index