

## Práctica 3 de Sistemas Operativos

Generated by Doxygen 1.8.14



# Contents

<b>1</b>	<b>Class Index</b>	<b>1</b>
1.1	Class List . . . . .	1
<b>2</b>	<b>File Index</b>	<b>3</b>
2.1	File List . . . . .	3
<b>3</b>	<b>Class Documentation</b>	<b>5</b>
3.1	Datos_Compartidos Struct Reference . . . . .	5
3.1.1	Detailed Description . . . . .	5
3.2	info Struct Reference . . . . .	5
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	msgbuf Struct Reference . . . . .	6
3.3.1	Detailed Description . . . . .	6
3.3.2	Member Data Documentation . . . . .	6
3.3.2.1	mtext . . . . .	7
3.3.2.2	mtype . . . . .	7
3.4	semun Union Reference . . . . .	7

<b>4</b>	<b>File Documentation</b>	<b>9</b>
4.1	aleat_num.c File Reference	9
4.1.1	Detailed Description	9
4.1.2	Function Documentation	9
4.1.2.1	aleat_num()	9
4.2	aleat_num.h File Reference	10
4.2.1	Detailed Description	10
4.2.2	Function Documentation	10
4.2.2.1	aleat_num()	10
4.3	cadena_montaje.c File Reference	11
4.3.1	Detailed Description	12
4.3.2	Macro Definition Documentation	12
4.3.2.1	ERROR	12
4.3.2.2	FILEKEY	13
4.3.2.3	KEY	13
4.3.2.4	OK	13
4.3.2.5	TAMANO_MENSAJE	13
4.3.3	Enumeration Type Documentation	13
4.3.3.1	Tipo_Mensaje	13
4.3.4	Function Documentation	13
4.3.4.1	inicializa_fichero_entrada()	14
4.3.4.2	main()	14
4.3.4.3	manejador()	14
4.3.4.4	numero_mensajes_pendientes()	15
4.3.4.5	proceso_A()	15
4.3.4.6	proceso_B()	16
4.3.4.7	proceso_C()	16
4.3.5	Variable Documentation	16
4.3.5.1	proceso_1_terminado	16
4.4	ejercicio2.c File Reference	17

4.4.1	Detailed Description	18
4.4.2	Macro Definition Documentation	18
4.4.2.1	FILEKEY	18
4.4.2.2	KEY	18
4.4.2.3	MAX_WAIT	18
4.4.2.4	MIN_WAIT	18
4.4.2.5	TAMANIO_NOMBRE	18
4.4.3	Function Documentation	19
4.4.3.1	ejecucionHijo()	19
4.4.3.2	main()	20
4.4.3.3	manejador()	20
4.4.3.4	reservashm()	21
4.5	ejercicio2_solved.c File Reference	21
4.5.1	Detailed Description	22
4.5.2	Macro Definition Documentation	22
4.5.2.1	FILEKEY	22
4.5.2.2	KEY	22
4.5.2.3	MAX_WAIT	23
4.5.2.4	MIN_WAIT	23
4.5.2.5	SEMAFORO_ENTRADA	23
4.5.2.6	SEMAFORO_SHM	23
4.5.2.7	TAMANIO_NOMBRE	23
4.5.3	Function Documentation	23
4.5.3.1	ejecucionHijo()	23
4.5.3.2	main()	24
4.5.3.3	manejador()	24
4.5.3.4	reservashm()	24
4.6	ejercicio3.c File Reference	25
4.6.1	Detailed Description	26
4.6.2	Macro Definition Documentation	26

4.6.2.1	ESPERA_INICIAL_CONSUMIDOR . . . . .	26
4.6.2.2	ESPERA_INICIAL_PRODUTOR . . . . .	26
4.6.2.3	FILEKEY . . . . .	26
4.6.2.4	KEY . . . . .	26
4.6.2.5	MAX_SLEEP_CONSUMIDOR . . . . .	27
4.6.2.6	MAX_SLEEP_PRODUTOR . . . . .	27
4.6.2.7	MIN_SLEEP_CONSUMIDOR . . . . .	27
4.6.2.8	MIN_SLEEP_PRODUTOR . . . . .	27
4.6.2.9	N . . . . .	27
4.6.2.10	NUMERO_SEMAFOROS . . . . .	27
4.6.3	Function Documentation . . . . .	27
4.6.3.1	consumidor() . . . . .	27
4.6.3.2	main() . . . . .	28
4.6.3.3	productor() . . . . .	28
4.6.3.4	reservashm() . . . . .	28
4.7	ejercicio4.c File Reference . . . . .	29
4.7.1	Detailed Description . . . . .	29
4.7.2	Macro Definition Documentation . . . . .	30
4.7.2.1	FILENAME . . . . .	30
4.7.3	Function Documentation . . . . .	30
4.7.3.1	main() . . . . .	30
4.7.3.2	primer_hilo() . . . . .	30
4.7.3.3	segundo_hilo() . . . . .	30
<b>Index</b>		<b>31</b>

# Chapter 1

## Class Index

### 1.1 Class List

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

<a href="#">Datos_Compartidos</a>	Estructura donde guardamos los valores para compartir entre procesos . . . . .	5
<a href="#">info</a>	Estructura que almacena la información de un usuario . . . . .	5
<a href="#">msgbuf</a>	Estructura que almacena los datos de los mensajes que van a ser enviados a la cola . . . . .	6
<a href="#">semun</a>	. . . . .	7





## Chapter 2

# File Index

### 2.1 File List

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

<a href="#">aleat_num.c</a>	Fichero con la función aleat_num . . . . .	9
<a href="#">aleat_num.h</a>	Fichero con la función aleat_num . . . . .	10
<a href="#">cadena_montaje.c</a>	Ejercicio 5 de la Práctica . . . . .	11
<a href="#">ejercicio2.c</a>	Ejercicio 2 de la Práctica . . . . .	17
<a href="#">ejercicio2_solved.c</a>	Ejercicio 2_solved de la Práctica . . . . .	21
<a href="#">ejercicio3.c</a>	Ejercicio 3 de la Práctica . . . . .	25
<a href="#">ejercicio4.c</a>	Ejercicio 4 de la Práctica . . . . .	29
<b>semaforos.h</b>	. . . . .	??



## Chapter 3

# Class Documentation

### 3.1 Datos\_Compartidos Struct Reference

Estructura donde guardamos los valores para compartir entre procesos.

#### Public Attributes

- int **contador**
- char **lista** [N]
- int **entrada**
- int **salida**

#### 3.1.1 Detailed Description

Estructura donde guardamos los valores para compartir entre procesos.

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

- [ejercicio3.c](#)

### 3.2 info Struct Reference

Estructura que almacena la información de un usuario.

#### Public Attributes

- char **nombre** [TAMANIO\_NOMBRE]
- int **id**

### 3.2.1 Detailed Description

Estructura que almacena la información de un usuario.

### 3.2.2 Member Data Documentation

#### 3.2.2.1 id

```
int info::id
```

Identificador del usuario

#### 3.2.2.2 nombre

```
char info::nombre
```

Nombre del usuario

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

- [ejercicio2.c](#)
- [ejercicio2\\_solved.c](#)

## 3.3 msgbuf Struct Reference

Estructura que almacena los datos de los mensajes que van a ser enviados a la cola.

### Public Attributes

- long [mtype](#)
- char [mtext](#) [[TAMANO\\_MENSAJE](#)]

### 3.3.1 Detailed Description

Estructura que almacena los datos de los mensajes que van a ser enviados a la cola.

### 3.3.2 Member Data Documentation

### 3.3.2.1 mtext

```
char msgbuf::mtext [TAMANO_MENSAJE]
```

Texto del mensaje a ser enviado

### 3.3.2.2 mtype

```
long msgbuf::mtype
```

Identificador del tipo de mensaje

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

- [cadena\\_montaje.c](#)

## 3.4 semun Union Reference

### Public Attributes

- int **val**
- struct semid\_ds \* **semstat**
- unsigned short \* **array**

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

- semaforos.c



## Chapter 4

# File Documentation

### 4.1 aleat\_num.c File Reference

Fichero con la función aleat\_num.

```
#include <stdlib.h>
```

#### Functions

- int [aleat\\_num](#) (int inf, int sup)  
*Genera un número aleatorio.*

#### 4.1.1 Detailed Description

Fichero con la función aleat\_num.

En este fichero definimos la función aleat\_num, ya que es utilizada en más de un fichero, y para así evitar reutilización del código

#### Author

José Manuel Chacón Aguilera y Miguel Arconada Manteca

#### Date

6-4-2018

#### 4.1.2 Function Documentation

##### 4.1.2.1 aleat\_num()

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

Genera un número aleatorio.

Esta funcion genera un número aleatorio entre dos especificados

**Parameters**

<i>inf</i>	Menor número posible
<i>sup</i>	Mayor número posible

**Returns**

Número generado

## 4.2 aleat\_num.h File Reference

Fichero con la función aleat\_num.

**Functions**

- int [aleat\\_num](#) (int inf, int sup)  
*Genera un número aleatorio.*

### 4.2.1 Detailed Description

Fichero con la función aleat\_num.

En este fichero definimos la función aleat\_num, ya que es utilizada en más de un fichero, y para así evitar reutilización del código

**Author**

José Manuel Chacón Aguilera y Miguel Arconada Manteca

**Date**

6-4-2018

### 4.2.2 Function Documentation

#### 4.2.2.1 aleat\_num()

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

Genera un número aleatorio.

Esta funcion genera un número aleatorio entre dos especificados



## Parameters

<i>inf</i>	Menor número posible
<i>sup</i>	Mayor número posible

## Returns

Número generado

## 4.3 cadena\_montaje.c File Reference

Ejercicio 5 de la Práctica.

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

## Classes

- struct [msgbuf](#)

*Estructura que almacena los datos de los mensajes que van a ser enviados a la cola.*

## Macros

- #define [TAMANO\\_MENSAJE](#) 2048
- #define [OK](#) 0
- #define [ERROR](#) 1
- #define [KEY](#) 1300
- #define [FILEKEY](#) "/bin/bash"

## Typedefs

- typedef struct [msgbuf](#) [Mensaje](#)

*Estructura que almacena los datos de los mensajes que van a ser enviados a la cola.*

## Enumerations

- enum [Tipo\\_Mensaje](#) { [PROCESO\\_AB](#) = 1, [PROCESO\\_BC](#) = 2 }

*Enumeracion utilizada para diferenciar los tipos de mensajes en la cola.*

## Functions

- void `inicializa_fichero_entrada` (char \*fichero\_entrada)  
*Función que inicializa el fichero que contiene los datos a ser modificados por los procesos.*
- int `numero_mensajes_pendientes` (int msqid, int \*cantidad)  
*Función que lee el numero de mensajes pendientes en la cola.*
- void `proceso_A` (char \*fichero\_entrada, int msqid, int child\_pid\_2)  
*Función de la ejecución del proceso A.*
- void `proceso_B` (int msqid)  
*Función de la ejecución del proceso b.*
- void `proceso_C` (char \*fichero\_salida, int msqid)  
*Función de la ejecución del proceso C.*
- void `manejador` (int senal)  
*Manejador de la senal para el proceso B.*
- int `main` (int argc, char \*\*argv)  
*Función principal del programa.*

## Variables

- int `proceso_1_terminado`

### 4.3.1 Detailed Description

Ejercicio 5 de la Práctica.

En este ejercicio creamos una serie de procesos hijos, que realizan la labor de diferentes partes de una cadena de montajes, en la que reciben mensajes del anterior, lo modifican y mandan este resultado al siguiente

#### Author

José Manuel Chacón Aguilera y Miguel Arconada Manteca

#### Date

17-4-2018

### 4.3.2 Macro Definition Documentation

#### 4.3.2.1 ERROR

```
#define ERROR 1
```

Macro devuelta si se encuentra algún error

#### 4.3.2.2 FILEKEY

```
#define FILEKEY "/bin/bash"
```

Filekey utilizada por ftok

#### 4.3.2.3 KEY

```
#define KEY 1300
```

Key utilizada por ftok

#### 4.3.2.4 OK

```
#define OK 0
```

Macro para representar ningún fallo

#### 4.3.2.5 TAMANO\_MENSAJE

```
#define TAMANO_MENSAJE 2048
```

Tamanoo del mensaje

### 4.3.3 Enumeration Type Documentation

#### 4.3.3.1 Tipo\_Mensaje

```
enum Tipo_Mensaje
```

Enumeracion utilizada para diferenciar los tipos de mensajes en la cola.

Enumerator

PROCESO_AB	Mensajes enviados por el proceso A al B
PROCESO_BC	Mensajes enviados por el proceso B al C

### 4.3.4 Function Documentation

#### 4.3.4.1 inicializa\_fichero\_entrada()

```
void inicializa_fichero_entrada (
    char * fichero_entrada )
```

Función que inicializa el fichero que contiene los datos a ser modificados por los procesos.

Esta función es llamada al principio de la ejecución, y prepara un fichero de datos con una cadena prolongada de caracteres, que van a ser leídos por los procesos en bloques de 2 kilobytes

##### Parameters

<i>fichero_entrada</i>	Path del fichero
------------------------	------------------

##### Returns

void

#### 4.3.4.2 main()

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

Función principal del programa.

Este programa coordina la ejecución de tres procesos hijos que ejecutan partes consecutivas de una cadena de montaje

##### Returns

0 si todo se ejecuta correctamente, y -1 en cualquier otro caso

#### 4.3.4.3 manejador()

```
void manejador (
    int senal )
```

Manejador de la señal para el proceso B.

Esta función se ejecuta cuando el proceso A manda la señal de que ha acabado al proceso B. Después, se modifica una flag global

##### Parameters

<i>senal</i>	Identificador de la señal recibida
--------------	------------------------------------

**Returns**

void

**4.3.4.4 numero\_mensajes\_pendientes()**

```
int numero_mensajes_pendientes (
    int msqid,
    int * cantidad )
```

Función que lee el numero de mensajes pendientes en la cola.

Esta función llama a las funcion msgctl, que devuelve mucha mucha informacion de la cola de mensajes, y devuelve simplemente el numero de mensajes pendientes

**Parameters**

<i>msqid</i>	Identificador de la cola de mensajes
<i>cantidad</i>	Entero donde guardar la cantidad

**Returns**

OK si la función se ejecuta de forma satisfactoria y ERROR si no

**4.3.4.5 proceso\_A()**

```
void proceso_A (
    char * fichero_entrada,
    int msqid,
    int child_pid_2 )
```

Función de la ejecución del proceso A.

Esta función engloba toda la ejecución del proceso A. Este lee de un fichero bloques de 2 kilobytes de caracteres, y los manda mediante mensajes al proceso B

**Parameters**

<i>fichero_entrada</i>	Path del fichero del que leer los datos
<i>msqid</i>	Identificador de la cola de mensajes
<i>child_pid_2</i>	PID del proceso B, necesario para mandarle una senal cuando acaba

**Returns**

void

#### 4.3.4.6 proceso\_B()

```
void proceso_B (  
    int msqid )
```

Función de la ejecución del proceso b.

Esta función engloba toda la ejecución del proceso B. Este lee los mensajes que le manda el proceso A, modificada cada caracter por el siguiente en el alfabeto, y manda este texto al proceso C

##### Parameters

<i>msqid</i>	Identificador de la cola de mensajes
--------------	--------------------------------------

##### Returns

void

#### 4.3.4.7 proceso\_C()

```
void proceso_C (  
    char * fichero_salida,  
    int msqid )
```

Función de la ejecución del proceso C.

Esta función engloba toda la ejecución del proceso C. Este recibe mensajes del proceso B con bloques de texto, y los vuelva a un fichero de texto

##### Parameters

<i>fichero_salida</i>	Path del fichero al que escribir los datos
<i>msqid</i>	Identificador de la cola de mensajes

##### Returns

void

### 4.3.5 Variable Documentation

#### 4.3.5.1 proceso\_1\_terminado

```
int proceso_1_terminado
```

Flag global que indica si el proceso A ha terminado

## 4.4 ejercicio2.c File Reference

Ejercicio 2 de la Práctica.

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

### Classes

- struct [info](#)

*Estructura que almacena la información de un usuario.*

### Macros

- #define [MAX\\_WAIT](#) 10
- #define [MIN\\_WAIT](#) 1
- #define [KEY](#) 1300
- #define [TAMANIO\\_NOMBRE](#) 80
- #define [FILEKEY](#) "/bin/bash"

### Typedefs

- typedef struct [info](#) [Informacion](#)

*Estructura que almacena la información de un usuario.*

### Functions

- void [manejador](#) (int senal)  
*Manejador de la senal.*
- int [reservashm](#) (int size, int key)  
*Función que solicita memoria compartida.*
- void [ejecucionHijo](#) ()  
*Función que ejecuta el hijo.*
- int [main](#) (int argc, char \*\*argv)  
*Función principal del programa.*
- void [ejecucionHijo](#) (int key)

#### 4.4.1 Detailed Description

Ejercicio 2 de la Práctica.

En este ejercicio creamos una serie de procesos que, mediante memoria compartida y la terminal preguntan al usuario una serie de nombres, y guardan la informacion de los usuarios

##### Author

José Manuel Chacón Aguilera y Miguel Arconada Manteca

##### Date

17-4-2018

#### 4.4.2 Macro Definition Documentation

##### 4.4.2.1 FILEKEY

```
#define FILEKEY "/bin/bash"
```

Filekey para ftok

##### 4.4.2.2 KEY

```
#define KEY 1300
```

Key para ftok

##### 4.4.2.3 MAX\_WAIT

```
#define MAX_WAIT 10
```

Máxima espera de los procesos (en segundos)

##### 4.4.2.4 MIN\_WAIT

```
#define MIN_WAIT 1
```

Míima espera de los procesos (en segundos)

##### 4.4.2.5 TAMANIO\_NOMBRE

```
#define TAMANIO_NOMBRE 80
```

Tamano del campo del nombre en memoria compartida



### 4.4.3 Function Documentation

#### 4.4.3.1 ejecucionHijo()

```
void ejecucionHijo ( )
```

Función que ejecuta el hijo.

Esta función engloba las acciones de los procesos hijos. Estas son preguntar por terminal el nombre y guardarlo en memoria compartida

**Parameters**

<i>size</i>	Tamano en bytes de la zona deseada
<i>key</i>	Key creada por ftok

**Returns**

Identificador de la zona de memoria compartida

**4.4.3.2 main()**

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

Función principal del programa.

Este programa coordina la ejecución de procesos hijos, solicitan y almacenan información de usuarios

**Returns**

0 si todo se ejecuta correctamente, y -1 en cualquier otro caso

**4.4.3.3 manejador()**

```
void manejador (
    int senal )
```

Manejador de la senal.

Esta función es llamada cuando se recibe la senal SIGUSR1, y simplemente permite, mediante la llamada a `pause()`, esperar a que reciba la senal, puesto que está vacía

**Parameters**

<i>senal</i>	Senal recibida
--------------	----------------

**Returns**

void

#### 4.4.3.4 reservashm()

```
int reservashm (
    int size,
    int key )
```

Función que solicita memoria compartida.

Esta función es llamada para solicitar memoria compartida. Si el key especificado ya ha sido creado, devuelve la zona, y si no la crea

##### Parameters

<i>size</i>	Tamano en bytes de la zona deseada
<i>key</i>	Key creada por ftok

##### Returns

Identificador de la zona de memoria compartida

## 4.5 ejercicio2\_solved.c File Reference

Ejercicio 2\_solved de la Práctica.

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

### Classes

- struct [info](#)  
*Estructura que almacena la información de un usuario.*

### Macros

- #define [MAX\\_WAIT](#) 10
- #define [MIN\\_WAIT](#) 1
- #define [KEY](#) 1300
- #define [TAMANIO\\_NOMBRE](#) 80
- #define [SEMAFORO\\_ENTRADA](#) 0
- #define [SEMAFORO\\_SHM](#) 1
- #define [FILEKEY](#) "/bin/bash"

## Typedefs

- typedef struct [info Informacion](#)  
*Estructura que almacena la información de un usuario.*

## Functions

- void [manejador](#) (int senal)  
*Manejador de la senal.*
- int [reservashm](#) (int size, int key)  
*Función que solicita memoria compartida.*
- void [ejecucionHijo](#) ()  
*Función que ejecuta el hijo.*
- int [main](#) (int argc, char \*\*argv)  
*Función principal del programa.*
- void [ejecucionHijo](#) (int key)

### 4.5.1 Detailed Description

Ejercicio 2\_solved de la Práctica.

En este ejercicio creamos una serie de procesos que, mediante memoria compartida y la terminal preguntan al usuario una serie de nombres, y guardan la informacion de los usuarios. Anadimos el uso de semáforos para regular y sincronizar los procesos

#### Author

José Manuel Chacón Aguilera y Miguel Arconada Manteca

#### Date

17-4-2018

### 4.5.2 Macro Definition Documentation

#### 4.5.2.1 FILEKEY

```
#define FILEKEY "/bin/bash"
```

Filekey para ftok

#### 4.5.2.2 KEY

```
#define KEY 1300
```

Key para ftok

#### 4.5.2.3 MAX\_WAIT

```
#define MAX_WAIT 10
```

Máxima espera de los procesos (en segundos)

#### 4.5.2.4 MIN\_WAIT

```
#define MIN_WAIT 1
```

Mínima espera de los procesos (en segundos)

#### 4.5.2.5 SEMAFORO\_ENTRADA

```
#define SEMAFORO_ENTRADA 0
```

Identificador del semaforo de la entrada por pantalla

#### 4.5.2.6 SEMAFORO\_SHM

```
#define SEMAFORO_SHM 1
```

Identificador del semaforo de memoria compartida

#### 4.5.2.7 TAMANIO\_NOMBRE

```
#define TAMANIO_NOMBRE 80
```

Tamaño del campo del nombre en memoria compartida

### 4.5.3 Function Documentation

#### 4.5.3.1 ejecucionHijo()

```
void ejecucionHijo ( )
```

Función que ejecuta el hijo.

Esta función engloba las acciones de los procesos hijos. Estas son preguntar por terminal el nombre y guardarlo en memoria compartida

##### Returns

Identificador de la zona de memoria compartida

#### 4.5.3.2 main()

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

Función principal del programa.

Este programa coordina la ejecución de procesos hijos, solicitan y almacenan información de usuarios. Añade el uso de semáforos.

##### Returns

0 si todo se ejecuta correctamente, y -1 en cualquier otro caso

#### 4.5.3.3 manejador()

```
void manejador (
    int senal )
```

Manejador de la señal.

Esta función es llamada cuando se recibe la señal SIGUSR1, y simplemente permite, mediante la llamada a `pause()`, esperar a que reciba la señal, puesto que está vacía.

##### Parameters

<i>senal</i>	Señal recibida
--------------	----------------

##### Returns

void

#### 4.5.3.4 reservashm()

```
int reservashm (
    int size,
    int key )
```

Función que solicita memoria compartida.

Esta función es llamada para solicitar memoria compartida. Si el `key` especificado ya ha sido creado, devuelve la zona, y si no la crea.

##### Parameters

<i>size</i>	Tamaño en bytes de la zona deseada
<i>key</i>	Key creada por <code>ftok</code>

## Returns

Identificador de la zona de memoria compartida

## 4.6 ejercicio3.c File Reference

Ejercicio 3 de la Práctica.

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

## Classes

- struct [Datos\\_Compartidos](#)

*Estructura donde guardamos los valores para compartir entre procesos.*

## Macros

- #define [KEY](#) 1300
- #define [FILEKEY](#) "/bin/bash"
- #define [N](#) 10
- #define [NUMERO\\_SEMAFOROS](#) 4
- #define [MIN\\_SLEEP\\_PRODUTOR](#) 1
- #define [MAX\\_SLEEP\\_PRODUTOR](#) 5
- #define [MIN\\_SLEEP\\_CONSUMIDOR](#) 1
- #define [MAX\\_SLEEP\\_CONSUMIDOR](#) 5
- #define [ESPERA\\_INICIAL\\_CONSUMIDOR](#) 10
- #define [ESPERA\\_INICIAL\\_PRODUTOR](#) 0

## Enumerations

- enum [semaforos](#) { [SEMAFORO\\_LISTA](#), [SEMAFORO\\_ENTRADA](#), [SEMAFORO\\_SALIDA](#), [SEMAFORO\\_←  
CONTADOR](#) }

*Enumeracion con los distintos semaforos que utilizamos.*

## Functions

- void [productor](#) (int key)  
*Funcion privada conteniendo la ejecución del productor.*
- void [consumidor](#) (int key)  
*Funcion privada conteniendo la ejecución del consumidor.*
- int [reservashm](#) (int size, int key)  
*Funcion privada para reservar memoria compartida.*
- int [main](#) ()  
*Funcion Principal del ejercicio.*

### 4.6.1 Detailed Description

Ejercicio 3 de la Práctica.

En este ejercicio se nos pide implementar el algoritmo de sincronización de procesos "Productor-Consumidor". Un proceso es el encargado de producir cierto recurso mientras el otro proceso lo consume, todo esto de forma correctamente sincronizada para asegurarnos de que se respeta la exclusión mutua en las zonas críticas y asegurándonos también de que no se produce ningún tipo de interbloqueo.

#### Author

José Manuel Chacón Aguilera y Miguel Arconada Manteca

#### Date

17-4-2018

### 4.6.2 Macro Definition Documentation

#### 4.6.2.1 ESPERA\_INICIAL\_CONSUMIDOR

```
#define ESPERA_INICIAL_CONSUMIDOR 10
```

Espera inicial realizada por el consumidor

#### 4.6.2.2 ESPERA\_INICIAL\_PRODUTOR

```
#define ESPERA_INICIAL_PRODUTOR 0
```

Espera inicial realizada por el productor

#### 4.6.2.3 FILEKEY

```
#define FILEKEY "/bin/bash"
```

FILEKEY necesario para ftok

#### 4.6.2.4 KEY

```
#define KEY 1300
```

KEY necesaria para el ftok



#### 4.6.2.5 MAX\_SLEEP\_CONSUMIDOR

```
#define MAX_SLEEP_CONSUMIDOR 5
```

Espera máxima del consumidor

#### 4.6.2.6 MAX\_SLEEP\_PRODUTOR

```
#define MAX_SLEEP_PRODUTOR 5
```

Espera máxima del productor

#### 4.6.2.7 MIN\_SLEEP\_CONSUMIDOR

```
#define MIN_SLEEP_CONSUMIDOR 1
```

Espera mínima del consumidor

#### 4.6.2.8 MIN\_SLEEP\_PRODUTOR

```
#define MIN_SLEEP_PRODUTOR 1
```

Espera mínima del productor

#### 4.6.2.9 N

```
#define N 10
```

Numero maximo de elementos que pueden estar listos para consumir a la vez

#### 4.6.2.10 NUMERO\_SEMAFOROS

```
#define NUMERO_SEMAFOROS 4
```

Numero de semaforos usados en el ejercicio

### 4.6.3 Function Documentation

#### 4.6.3.1 consumidor()

```
void consumidor (  
    int key )
```

Funcion privada conteniendo la ejecución del consumidor.

**Parameters**

<i>key</i>	KEY para la memoria compartida
------------	--------------------------------

**Returns**

void

**4.6.3.2 main()**

```
int main ( )
```

Funcion Principal del ejercicio.

Crea los procesos hijos necesarios para implementar el productor-consumidor pedido en la documentación de la práctica

**Returns**

0 si todo se ejecuta correctamente, y -1 en cualquier otro caso

**4.6.3.3 productor()**

```
void productor (
    int key )
```

Funcion privada conteniendo la ejecución del productor.

**Parameters**

<i>key</i>	KEY para la memoria compartida
------------	--------------------------------

**Returns**

void

**4.6.3.4 reservashm()**

```
int reservashm (
    int size,
    int key )
```

Funcion privada para reservar memoria compartida.

La llamamos en el Main cada vez que necesitamos memoria compartida para la comunicación de procesos.

## Parameters

<i>size</i>	tamano de la memoria deseado
<i>key</i>	clave de la memoria compartida

## Returns

El identificador de la memoria compartida creada

## 4.7 ejercicio4.c File Reference

Ejercicio 4 de la Práctica.

```
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>
#include <sys/types.h>
#include <sys/shm.h>
#include <sys/mman.h>
#include <sys/wait.h>
#include <sys/stat.h>
#include <sys/ipc.h>
#include <fcntl.h>
#include <signal.h>
#include <string.h>
#include <unistd.h>
#include "aleat_num.h"
```

## Macros

- `#define` [FILENAME](#) "files/ej4/fichero\_ej4"

## Functions

- `void *` [primer\\_hilo](#) ()  
*Ejecución del primer hilo, escribe números aleatorios en un fichero de texto.*
- `void *` [segundo\\_hilo](#) ()  
*Ejecución del Segundo hilo, mapea la memoria del fichero escrito por el primer hilo y modifica el.*
- `int` [main](#) ()  
*Función Main del ejercicio, invoca a dos hilos y posteriormente los ejecuta con las funciones descritas en este mismo archivo.*

### 4.7.1 Detailed Description

Ejercicio 4 de la Práctica.

En este ejercicio se nos pide trabajar con la funcion de C mmap que es útil para mapear memoria y acceder desde otros procesos a la misma. Un hilo se comunica en este ejercicio con otro hilo mediante el uso de dicha función.

## Author

José Manuel Chacón Aguilera y Miguel Arconada Manteca

## Date

17-4-2018

## 4.7.2 Macro Definition Documentation

### 4.7.2.1 FILENAME

```
#define FILENAME "files/ej4/fichero_ej4"
```

Nombre del Fichero donde se genera

## 4.7.3 Function Documentation

### 4.7.3.1 main()

```
int main ( )
```

Función Main del ejercicio, invoca a dos hilos y posteriormente los ejecuta con las funciones descritas en este mismo archivo.

#### Returns

0 si todo se ejecuta correctamente, y -1 en cualquier otro caso

### 4.7.3.2 primer\_hilo()

```
void * primer_hilo ( )
```

Ejecución del primer hilo, escribe números aleatorios en un fichero de texto.

#### Returns

void

### 4.7.3.3 segundo\_hilo()

```
void * segundo_hilo ( )
```

Ejecución del Segundo hilo, mapea la memoria del fichero escrito por el primer hilo y modifica el.

#### Returns

void

# Index

- aleat\_num
  - aleat\_num.c, 9
  - aleat\_num.h, 10
- aleat\_num.c, 9
  - aleat\_num, 9
- aleat\_num.h, 10
  - aleat\_num, 10
- cadena\_montaje.c, 11
  - ERROR, 12
  - FILEKEY, 12
  - inicializa\_fichero\_entrada, 13
  - KEY, 13
  - main, 14
  - manejador, 14
  - numero\_mensajes\_pendientes, 15
  - OK, 13
  - proceso\_1\_terminado, 16
  - proceso\_A, 15
  - proceso\_B, 15
  - proceso\_C, 16
  - TAMANO\_MENSAJE, 13
  - Tipo\_Mensaje, 13
- consumidor
  - ejercicio3.c, 27
- Datos\_Compartidos, 5
- ERROR
  - cadena\_montaje.c, 12
- ESPERA\_INICIAL\_CONSUMIDOR
  - ejercicio3.c, 26
- ESPERA\_INICIAL\_PRODUCTOR
  - ejercicio3.c, 26
- ejecucionHijo
  - ejercicio2.c, 19
  - ejercicio2\_solved.c, 23
- ejercicio2.c, 17
  - ejecucionHijo, 19
  - FILEKEY, 18
  - KEY, 18
  - MAX\_WAIT, 18
  - MIN\_WAIT, 18
  - main, 20
  - manejador, 20
  - reservashm, 20
  - TAMANIO\_NOMBRE, 18
- ejercicio2\_solved.c, 21
  - ejecucionHijo, 23
  - FILEKEY, 22
- KEY, 22
- MAX\_WAIT, 22
- MIN\_WAIT, 23
- main, 23
- manejador, 24
- reservashm, 24
- SEMAFORO\_ENTRADA, 23
- SEMAFORO\_SHM, 23
- TAMANIO\_NOMBRE, 23
- ejercicio3.c, 25
  - consumidor, 27
  - ESPERA\_INICIAL\_CONSUMIDOR, 26
  - ESPERA\_INICIAL\_PRODUCTOR, 26
  - FILEKEY, 26
  - KEY, 26
  - MAX\_SLEEP\_CONSUMIDOR, 26
  - MAX\_SLEEP\_PRODUCTOR, 27
  - MIN\_SLEEP\_CONSUMIDOR, 27
  - MIN\_SLEEP\_PRODUCTOR, 27
  - main, 28
  - N, 27
  - NUMERO\_SEMAFOROS, 27
  - productor, 28
  - reservashm, 28
- ejercicio4.c, 29
  - FILENAME, 30
  - main, 30
  - primer\_hilo, 30
  - segundo\_hilo, 30
- FILEKEY
  - cadena\_montaje.c, 12
  - ejercicio2.c, 18
  - ejercicio2\_solved.c, 22
  - ejercicio3.c, 26
- FILENAME
  - ejercicio4.c, 30
- id
  - info, 6
- info, 5
  - id, 6
  - nombre, 6
- inicializa\_fichero\_entrada
  - cadena\_montaje.c, 13
- KEY
  - cadena\_montaje.c, 13
  - ejercicio2.c, 18
  - ejercicio2\_solved.c, 22

- [ejercicio3.c, 26](#)
- MAX\_SLEEP\_CONSUMIDOR
  - [ejercicio3.c, 26](#)
- MAX\_SLEEP\_PRODUTOR
  - [ejercicio3.c, 27](#)
- MAX\_WAIT
  - [ejercicio2.c, 18](#)
  - [ejercicio2\\_solved.c, 22](#)
- MIN\_SLEEP\_CONSUMIDOR
  - [ejercicio3.c, 27](#)
- MIN\_SLEEP\_PRODUTOR
  - [ejercicio3.c, 27](#)
- MIN\_WAIT
  - [ejercicio2.c, 18](#)
  - [ejercicio2\\_solved.c, 23](#)
- main
  - [cadena\\_montaje.c, 14](#)
  - [ejercicio2.c, 20](#)
  - [ejercicio2\\_solved.c, 23](#)
  - [ejercicio3.c, 28](#)
  - [ejercicio4.c, 30](#)
- manejador
  - [cadena\\_montaje.c, 14](#)
  - [ejercicio2.c, 20](#)
  - [ejercicio2\\_solved.c, 24](#)
- msgbuf, [6](#)
  - [mtext, 6](#)
  - [mtype, 7](#)
- mtext
  - [msgbuf, 6](#)
- mtype
  - [msgbuf, 7](#)
- N
  - [ejercicio3.c, 27](#)
- NUMERO\_SEMAFOROS
  - [ejercicio3.c, 27](#)
- nombre
  - [info, 6](#)
- numero\_mensajes\_pendientes
  - [cadena\\_montaje.c, 15](#)
- OK
  - [cadena\\_montaje.c, 13](#)
- primer\_hilo
  - [ejercicio4.c, 30](#)
- proceso\_1\_terminado
  - [cadena\\_montaje.c, 16](#)
- proceso\_A
  - [cadena\\_montaje.c, 15](#)
- proceso\_B
  - [cadena\\_montaje.c, 15](#)
- proceso\_C
  - [cadena\\_montaje.c, 16](#)
- productor
  - [ejercicio3.c, 28](#)
- reservashm
  - [ejercicio2.c, 20](#)
  - [ejercicio2\\_solved.c, 24](#)
  - [ejercicio3.c, 28](#)
- SEMAFORO\_ENTRADA
  - [ejercicio2\\_solved.c, 23](#)
- SEMAFORO\_SHM
  - [ejercicio2\\_solved.c, 23](#)
- segundo\_hilo
  - [ejercicio4.c, 30](#)
- semun, [7](#)
- TAMANIO\_NOMBRE
  - [ejercicio2.c, 18](#)
  - [ejercicio2\\_solved.c, 23](#)
- TAMANO\_MENSAJE
  - [cadena\\_montaje.c, 13](#)
- Tipo\_Mensaje
  - [cadena\\_montaje.c, 13](#)