# Practica 2

Generado por Doxygen 1.8.11

# Índice general

# Capítulo 1

# Indice de archivos

## 1.1. Lista de archivos

Lista de todos los archivos documentados y con descripciones breves:

# Capítulo 2

# Documentación de archivos

## 2.1. Referencia del Archivo ejercicio2.c

Ejercicio2.

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <signal.h>
#include <sys/wait.h>
```

Dependencia gráfica adjunta para ejercicio2.c:

## 2.2. Referencia del Archivo ejercicio4.c

Ejercicio4.

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <signal.h>
#include <sys/wait.h>
```

Dependencia gráfica adjunta para ejercicio4.c:

### 'defines'

- #define TAM 5

  *Numero de hijos a crear.*

### Funciones

- void captura ()

  *Funcion capturadora de la señal.*
- int main ()

  *Main del ejercicio 4.*

#### 2.2.1. Descripción detallada

Ejercicio4.

**Autor**

Lucia Rivas Molina
Daniel Santo-Tomas Lopez

**Fecha**

31/03/2018

#### 2.2.2. Documentación de las funciones

##### 2.2.2.1. int main ( void )

Main del ejercicio 4.

**Devuelve**

EXIT_SUCCESS o EXIT_FAILURE

## 2.3. Referencia del Archivo ejercicio6a.c

Ejercicio6a.

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/wait.h>
#include <signal.h>
#include <time.h>
```

Dependencia gráfica adjunta para ejercicio6a.c:

### 'defines'

- #define NUM_PROC 5

  *Numeros que cuenta el proceso.*
- #define SEC 40

  *Segundos que espera a la alarma.*

### Funciones

- int main (void)

  *Main del ejercicio 6a.*

### 2.3.1. Descripción detallada

Ejercicio6a.

**Autor**

Lucia Rivas Molina
Daniel Santo-Tomas Lopez

**Fecha**

31/03/2018

### 2.3.2. Documentación de las funciones

#### 2.3.2.1. int main ( void )

Main del ejercicio 6a.

**Devuelve**

EXIT_SUCCESS o EXIT_FAILURE

## 2.4. Referencia del Archivo ejercicio6b.c

Ejercicio6b.

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/wait.h>
#include <signal.h>
#include <time.h>
```

Dependencia gráfica adjunta para ejercicio6b.c:

### 'defines'

- #define NUM_PROC 5

  *Contador del hijo.*
- #define SEC 40

  *Segundos que espera a la alarma.*

### Funciones

- void captura (int sennal)

  *Funcion captura de la señal.*
- int main (void)

  *Main del ejercicio 6b.*

### 2.4.1. Descripción detallada

Ejercicio6b.

**Autor**

Lucia Rivas Molina
Daniel Santo-Tomas Lopez

**Fecha**

31/03/2018

### 2.4.2. Documentación de las funciones

#### 2.4.2.1. void captura ( int *sennal* )

Funcion captura de la señal.

**Parámetros**

| | |
|---|---|
| *sennal* | Señal recibida |

#### 2.4.2.2. int main ( void )

Main del ejercicio 6b.

**Devuelve**

EXIT_SUCCESS o EXIT_FAILURE

## 2.5. Referencia del Archivo ejercicio9.c

Ejercicio9.

```
#include <unistd.h>
#include <sys/wait.h>
#include <signal.h>
#include <time.h>
#include "semaforos.h"
```

Dependencia gráfica adjunta para ejercicio9.c:

### 'defines'

- #define CAJAS 5

  *numero de cajas del supermercado*
- #define SEMAFOROS CAJAS+1

  *numero de semaforos necesarios Necesitamos un semaforo para el archivo de cada caja y uno más para el archivo info.txt que guarda la señal mandada y la caja que la envía*
- #define OPERAC 50

  *numero de corbos a clientes que realiza cada caja*
- #define ALEAT aleat_num(0, 300)

  *establece un numero aleatorio entre 0 y 300 para la escritura de los archivos clientesCaja.txt*
- #define SECS aleat_num(1,5)

  *establece un numero aleatorio entre 0 y 5 para los segundos del sleep de cada caja*
- #define SEMKEY 75798

  *key para los semaforos*

### Funciones

- int aleat_num (int inf, int sup)

  *Devuelve un numero aleatorio entre inf y sup En caso de pasar un numero negativo se cambia de signo En caso de que sup sea menor inf se permutan.*
- int leer (char ∗archivo)

  *Lee el numero guardado en el fichero archivo y lo devuelve. En caso de error devuelve ERROR.*
- int escribir (char ∗archivo, int cantidad)

  *Escribe el numero cantidad en el fichero archivo y devuelve, en caso de error ERROR y si sale bien OK.*
- void captura (int signal)

  *Funcion capturadora de la señal. Si la señal capturada es SIGUSR1 el padre retira 900 euros. Si la señal es SIGUSR2 el padre retira todo el dinero pues el hijo ha terminado.*
- int vaciar (char ∗archivo)

  *Funcion que vacía un archivo abriendolo con "w" y cerrandolo.*
- int caja (char ∗clientesCaja, char ∗cajaTotal, char ∗info, int i, int semid)

  *Función que contiene toda la estructura de lo que realiza cada hijo.*
- int main (void)

  *Main del ejercicio 9.*

## 2.5.1. Descripción detallada

Ejercicio9.

**Autor**

Lucia Rivas Molina
Daniel Santo-Tomas Lopez

**Fecha**

31/03/2018

## 2.5.2. Documentación de las funciones

### 2.5.2.1. int aleat_num ( int *inf*, int *sup* )

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

**Parámetros**

| | |
|---|---|
| *inf* | minimo numero aleatorio que puede salir |
| *sup* | maximo numero aleatorio que puede salir |

**Devuelve**

el numero aleatorio

#### 2.5.2.2. int caja ( char * *clientesCaja,* char * *cajaTotal,* char * *info,* int *i,* int *semid* )

Función que contiene toda la estructura de lo que realiza cada hijo.

En primer lugar, el hijo abre clientesCaja.txt y va leyendo lo que paga cada cliente, guardándolo en dinero. Luego, hace un down del semaforo de cajaTotal.txt para reescribir el total y el up. Si el total es <=1000, hace un down del semaforo de info.txt para escribir SIGUSR1 hace el up, un kill y un sleep. Cuando ha terminado, sale del while, cierra el fichero clientesCaja.txt y hace un down de info.txt para mandarle al padre SIGUSR2 y hace un exit.

**Parámetros**

| | |
|---|---|
| *clientesCaja* | contiene clientesCaja.txt del que caja hijo lee lo que paga el cliente |
| *cajaTotal* | contiene cajaTotal.txt donde cada hijo guarda su total sumado |
| *info* | contiene info.txt donde cada hijo escribe la señal que manda al padre y que hjo es |
| *i* | int que indica que hijo/caja es |
| *semid* | el semaforo |

**Devuelve**

ERROR en caso de error, sino OK

#### 2.5.2.3. void captura ( int *signal* )

Funcion capturadora de la señal. Si la señal capturada es SIGUSR1 el padre retira 900 euros. Si la señal es SIGUSR2 el padre retira todo el dinero pues el hijo ha terminado.

**Parámetros**

| | |
|---|---|
| *signal* | señal que captura |

**Devuelve**

void

#### 2.5.2.4. int escribir ( char * *archivo,* int *cantidad* )

Escribe el numero cantidad en el fichero archivo y devuelve, en caso de error ERROR y si sale bien OK.

**Parámetros**

| | |
|---|---|
| *archivo* | nombre del fichero a escribir |
| *cantidad* | a escribir |

**Devuelve**

OK o ERROR

#### 2.5.2.5. int leer ( char * *archivo* )

Lee el numero guardado en el fichero archivo y lo devuelve. En caso de error devuelve ERROR.

**Parámetros**

| | |
|---|---|
| *archivo* | nombre del fichero a leer |

**Devuelve**

el numero leido o ERROR

#### 2.5.2.6. int main ( void )

Main del ejercicio 9.

**Devuelve**

EXIT_FAILURE o EXIT_SUCCESS

#### 2.5.2.7. int vaciar ( char * *archivo* )

Funcion que vacía un archivo abriendolo con "w" y cerrandolo.

**Parámetros**

| | |
|---|---|
| *archivo* | array de char que contiene el fichero a vaciar |

**Devuelve**

ERROR o OK

## 2.6. Referencia del Archivo semaforos.c

Biblioteca de semaforos.

```
#include "semaforos.h"
```
Dependencia gráfica adjunta para semaforos.c:

## Funciones

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

  *Inicializa los semaforos inficados.*
- int Borrar_Semaforo (int semid)

  *Elimina los semaforos indicados.*
- int Crear_Semaforo (key_t key, int size, int *semid)

  *Crea los semaforos indicados.*
- int Down_Semaforo (int id, int num_sem, int undo)

  *Hace un down de un semaforo.*
- int DownMultiple_Semaforo (int id, int size, int undo, int *active)

  *Hace un down de un grupo de semaforos.*
- int Up_Semaforo (int id, int num_sem, int undo)

  *Hace un up de un semaforo.*
- int UpMultiple_Semaforo (int id, int size, int undo, int *active)

  *Hace un up de un grupo de semaforos.*

### 2.6.1. Descripción detallada

Biblioteca de semaforos.

Biblioteca de semaforos para las prácticas de SOPER

**Autor**

Lucia Rivas Molina lucia.rivasmolina@estudiante.uam.es
Daniel Santo-Tomas Lopez daniel.santo-tomas@estudiante.uam.es

**Fecha**

20/03/2018

### 2.6.2. Documentación de las funciones

#### 2.6.2.1. int Borrar_Semaforo ( int *semid* )

Elimina los semaforos indicados.

**Parámetros**

| | |
|---|---|
| *semid* | : identificador del grupo de semaforos |

**Devuelve**

OK / ERROR

#### 2.6.2.2. int Crear_Semaforo ( key_t *key,* int *size,* int * *semid* )

Crea los semaforos indicados.

**Parámetros**

| | |
|---|---|
| *key* | : clave precompartida del semaforo |
| *size* | : numero de semaforos a crear |
| *semid* | : identificador del grupo de semaforos |

**Devuelve**

OK / ERROR

#### 2.6.2.3. int Down_Semaforo ( int *id,* int *num_sem,* int *undo* )

Hace un down de un semaforo.

**Parámetros**

| | |
|---|---|
| *id* | : identificador del grupo de semaforos |
| *num_sem* | : numero del semaforo del cual hacemos el down |
| *undo* | : flags |

**Devuelve**

OK / ERROR

#### 2.6.2.4. int DownMultiple_Semaforo ( int *id,* int *size,* int *undo,* int * *active* )

Hace un down de un grupo de semaforos.

**Parámetros**

| | |
|---|---|
| *id* | : identificador del grupo de semaforos |
| *size* | : numero de semaforos para hacer un down |
| *undo* | : flags |
| *active* | : semaforos involucrados |

**Devuelve**

OK / ERROR

#### 2.6.2.5. int Inicializar_Semaforo ( int *semid,* unsigned short * *array* )

Inicializa los semaforos inficados.

**Parámetros**

| | |
|---|---|
| *semid* | : identificador del grupo de semaforos |
| *array* | : valores iniciales del semaforo |

**Devuelve**

OK / ERROR

#### 2.6.2.6. int Up_Semaforo ( int *id,* int *num_sem,* int *undo* )

Hace un up de un semaforo.

**Parámetros**

| | |
|---|---|
| *id* | : identificador del grupo de semaforos |
| *num_sem* | : numero del semaforo del cual hacemos el down |
| *undo* | : flags |

**Devuelve**

OK / ERROR

#### 2.6.2.7. int UpMultiple_Semaforo ( int *id,* int *size,* int *undo,* int * *active* )

Hace un up de un grupo de semaforos.

**Parámetros**

| | |
|---|---|
| *id* | : identificador del grupo de semaforos |
| *size* | : numero de semaforos para hacer un down |
| *undo* | : flags |
| *active* | : semaforos involucrados |

**Devuelve**

OK / ERROR

# Índice alfabético