Práctica 3 de Sistemas Operativos

Generated by Doxygen 1.8.14

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 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

inf	Menor número posible
sup	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

inf	Menor número posible
sup	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 ejecucion, y prepara un fichero de datos con una cadena prolongada de caracteres, que van a ser leidos por los procesos en bloques de 2 kilobytes

Parameters

fichero_entrada	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 senal para el proceso B.

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

Parameters

senal	Identificador de la senal 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 simplemetne el numero de mensajes pendientes

Parameters

msqid	Identificador de la cola de mensajes
cantidad	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

fichero_entrada	Path del fichero del que leer los datos
msqid	Identificador de la cola de mensajes
child_pid_2	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

msqid	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

fichero_salida	Path del fichero al que escribir los datos
msqid	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

size	Tamano en bytes de la zona deseada
key	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

senal	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

size	Tamano en bytes de la zona deseada
key	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
```

Tamano 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. Anade el uso de semáfoross

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 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

senal	Senal 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

size	Tamano en bytes de la zona deseada
key	Key creada por ftok

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_PRODUCTOR 1
- #define MAX_SLEEP_PRODUCTOR 5
- #define MIN_SLEEP_CONSUMIDOR 1
- #define MAX_SLEEP_CONSUMIDOR 5
- #define ESPERA_INICIAL_CONSUMIDOR 10
- #define ESPERA_INICIAL_PRODUCTOR 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 aseguranrnos de que se respeta la exclusión mutua en las zonas críticas y asegurandones tambien de que no se produce ningun 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_PRODUCTOR

```
#define ESPERA_INICIAL_PRODUCTOR 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_PRODUCTOR

```
#define MAX_SLEEP_PRODUCTOR 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_PRODUCTOR

```
#define MIN_SLEEP_PRODUCTOR 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

key	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

key	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

size	tamano de la memoria deseado
key	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