

## Primera práctica de Sistemas Operativos

Generado por Doxygen 1.8.14



# Índice general

<b>1</b>	<b>Índice de clases</b>	<b>1</b>
1.1	Lista de clases . . . . .	1
<b>2</b>	<b>Indice de archivos</b>	<b>3</b>
2.1	Lista de archivos . . . . .	3
<b>3</b>	<b>Documentación de las clases</b>	<b>5</b>
3.1	Referencia de la Estructura <code>_estructura</code> . . . . .	5
3.1.1	Descripción detallada . . . . .	5
3.1.2	Documentación de los datos miembro . . . . .	5
3.1.2.1	<code>cadena</code> . . . . .	5
3.1.2.2	<code>entero</code> . . . . .	6
3.2	Referencia de la Estructura <code>param</code> . . . . .	6
3.2.1	Descripción detallada . . . . .	6
3.2.2	Documentación de los datos miembro . . . . .	6
3.2.2.1	<code>dim</code> . . . . .	6
3.2.2.2	<code>id</code> . . . . .	6
3.2.2.3	<code>matriz</code> . . . . .	6
3.2.2.4	<code>mult</code> . . . . .	6

<b>4 Documentación de archivos</b>	<b>7</b>
4.1 Referencia del Archivo ejercicio12a.c	7
4.1.1 Descripción detallada	8
4.1.2 Documentación de los 'defines'	8
4.1.2.1 NUMERO_PROCESOS	8
4.1.3 Documentación de los 'typedefs'	8
4.1.3.1 estructura	8
4.1.4 Documentación de las funciones	8
4.1.4.1 calcula_primos()	8
4.1.4.2 main()	9
4.2 Referencia del Archivo ejercicio12b.c	9
4.2.1 Descripción detallada	10
4.2.2 Documentación de los 'defines'	10
4.2.2.1 NUMERO_HILOS	10
4.2.3 Documentación de los 'typedefs'	10
4.2.3.1 estructura	11
4.2.4 Documentación de las funciones	11
4.2.4.1 calcula_primos()	11
4.2.4.2 main()	11
4.3 Referencia del Archivo ejercicio13.c	12
4.3.1 Descripción detallada	12
4.3.2 Documentación de los 'defines'	13
4.3.2.1 ARRAY_SIZE	13
4.3.2.2 BUFFER_SIZE	13
4.3.3 Documentación de los 'typedefs'	13
4.3.3.1 Param	13
4.3.4 Documentación de las funciones	13
4.3.4.1 calcula_matriz()	13
4.3.4.2 main()	14
4.4 Referencia del Archivo ejercicio4a.c	14

4.4.1	Descripción detallada . . . . .	14
4.4.2	Documentación de los 'defines' . . . . .	15
4.4.2.1	NUM_PROC . . . . .	15
4.5	Referencia del Archivo ejercicio4b.c . . . . .	15
4.5.1	Descripción detallada . . . . .	15
4.5.2	Documentación de los 'defines' . . . . .	15
4.5.2.1	NUM_PROC . . . . .	16
4.6	Referencia del Archivo ejercicio5a.c . . . . .	16
4.6.1	Descripción detallada . . . . .	16
4.6.2	Documentación de los 'defines' . . . . .	16
4.6.2.1	NUM_PROC . . . . .	16
4.7	Referencia del Archivo ejercicio5b.c . . . . .	17
4.7.1	Descripción detallada . . . . .	17
4.7.2	Documentación de los 'defines' . . . . .	17
4.7.2.1	NUM_PROC . . . . .	17
4.8	Referencia del Archivo ejercicio6.c . . . . .	18
4.8.1	Descripción detallada . . . . .	18
4.8.2	Documentación de los 'defines' . . . . .	18
4.8.2.1	MAX_CAD . . . . .	18
4.9	Referencia del Archivo ejercicio8.c . . . . .	19
4.9.1	Descripción detallada . . . . .	19
4.9.2	Documentación de los 'defines' . . . . .	19
4.9.2.1	MAX_PATH . . . . .	19
4.10	Referencia del Archivo ejercicio9.c . . . . .	20
4.10.1	Descripción detallada . . . . .	20
4.10.2	Documentación de los 'defines' . . . . .	20
4.10.2.1	BUFFER_SIZE . . . . .	21
4.10.2.2	ESCRITURA . . . . .	21
4.10.2.3	LECTURA . . . . .	21
4.10.2.4	MENSAJE_SIZE . . . . .	21
4.10.3	Documentación de las funciones . . . . .	21
4.10.3.1	factorial() . . . . .	21
4.10.3.2	main() . . . . .	22



# Capítulo 1

## Índice de clases

### 1.1. Lista de clases

Lista de las clases, estructuras, uniones e interfaces con una breve descripción:

<a href="#">_estructura</a>	Estructura generada al principio de la ejecución . . . . .	5
<a href="#">param</a>	Estructura de parámetros de un hilo . . . . .	6





## Capítulo 2

# Indice de archivos

### 2.1. Lista de archivos

Lista de todos los archivos documentados y con descripciones breves:

<a href="#">ejercicio12a.c</a>	
Ejercicio para la comparación de tiempo: procesos . . . . .	7
<a href="#">ejercicio12b.c</a>	
Ejercicio para la comparación de tiempo: hilos . . . . .	9
<a href="#">ejercicio13.c</a>	
Ejercicio sobre paso de parámetros en funciones . . . . .	12
<a href="#">ejercicio4a.c</a>	
Ejercicio 4a de la Práctica . . . . .	14
<a href="#">ejercicio4b.c</a>	
Ejercicio 4b de la Práctica . . . . .	15
<a href="#">ejercicio5a.c</a>	
Ejercicio 5a de la Práctica . . . . .	16
<a href="#">ejercicio5b.c</a>	
Ejercicio 5b de la Práctica . . . . .	17
<a href="#">ejercicio6.c</a>	
Ejercicio 6 de la Práctica . . . . .	18
<a href="#">ejercicio8.c</a>	
Ejercicio 8 de la Práctica . . . . .	19
<a href="#">ejercicio9.c</a>	
Ejercicio para la comunicacion entre procesos mediante tuberias . . . . .	20



## Capítulo 3

# Documentación de las clases

### 3.1. Referencia de la Estructura `_estructura`

estructura generada al principio de la ejecución

#### Atributos públicos

- `char * cadena`
- `int * entero`

#### 3.1.1. Descripción detallada

estructura generada al principio de la ejecución

Esta estructura almacena una cadena de 100 caracteres y un entero. Se usa para comparar el uso de memoria entre los hilos y los procesos

#### 3.1.2. Documentación de los datos miembro

##### 3.1.2.1. `cadena`

```
char * _estructura::cadena
```

Cadena de 100 caracteres

### 3.1.2.2. entero

```
int * _estructura::entero
```

Puntero a un entero

La documentación para esta estructura fue generada a partir de los siguientes ficheros:

- [ejercicio12a.c](#)
- [ejercicio12b.c](#)

## 3.2. Referencia de la Estructura param

Estructura de parámetros de un hilo.

### Atributos públicos

- int \*\* [matriz](#)
- int [mult](#)
- int [dim](#)
- int [id](#)

### 3.2.1. Descripción detallada

Estructura de parámetros de un hilo.

Esta estructura almacena la información necesaria para los hilos en este ejercicio

### 3.2.2. Documentación de los datos miembro

#### 3.2.2.1. dim

```
int param::dim
```

Dimension de las matrices

#### 3.2.2.2. id

```
int param::id
```

Identificador del hilo

#### 3.2.2.3. matriz

```
int** param::matriz
```

Matriz

#### 3.2.2.4. mult

```
int param::mult
```

Multiplicador

La documentación para esta estructura fue generada a partir del siguiente fichero:

- [ejercicio13.c](#)

## Capítulo 4

# Documentación de archivos

### 4.1. Referencia del Archivo ejercicio12a.c

Ejercicio para la comparación de tiempo: procesos.

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

#### Clases

- struct `_estructura`  
*estructura generada al principio de la ejecución*

#### defines

- #define `NUMERO_PROCESOS` 100

#### typedefs

- typedef struct `_estructura` `estructura`  
*estructura generada al principio de la ejecución*

#### Funciones

- void `calcula_primos` (int N)  
*calcula los n primos numeros primos*
- int `main` (int argc, char \*\*argv)  
*Función principal del programa.*

#### 4.1.1. Descripción detallada

Ejercicio para la comparación de tiempo: procesos.

Este fichero, en conjunto con el [ejercicio12b.c](#), sirven para comparar la eficiencia de los threads comparados con los procesos

##### Autor

Miguel Arconada Manteca y José Manuel Chacón Aguilera

##### Fecha

8-3-2018

#### 4.1.2. Documentación de los 'defines'

##### 4.1.2.1. NUMERO\_PROCESOS

```
#define NUMERO_PROCESOS 100
```

Numero total de procesos que queremos crear

#### 4.1.3. Documentación de los 'typedefs'

##### 4.1.3.1. estructura

```
typedef struct _estructura estructura
```

estructura generada al principio de la ejecución

Esta estructura almacena una cadena de 100 caracteres y un entero. Se usa para comparar el uso de memoria entre los hilos y los procesos

#### 4.1.4. Documentación de las funciones

##### 4.1.4.1. calcula\_primos()

```
void calcula_primos (  
    int N )
```

calcula los n primos numeros primos

Esta funcion calcula los n primero numeros primos

**Parámetros**

<i>N</i>	numero de primos a calcular
----------	-----------------------------

**Devuelve**

void

**4.1.4.2. main()**

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

Función principal del programa.

Este programa lanza 100 procesos simultáneamente, en los que se calculan los N primeros primos

**Parámetros**

<i>argc</i>	numero de parametros de entrada
<i>argv</i>	array de cada parametro. El primero es el nombre del programa, y el segundo es N

**Devuelve**

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

**4.2. Referencia del Archivo ejercicio12b.c**

Ejercicio para la comparación de tiempo: hilos.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <pthread.h>
#include <time.h>
#include <math.h>
```

**Clases**

- struct [\\_estructura](#)

*estructura generada al principio de la ejecución*

## defines

- `#define NUMERO_HILOS 100`

## typedefs

- `typedef struct _estructura estructura`  
*estructura generada al principio de la ejecución*

## Funciones

- `void * calcula_primos (void *arg)`  
*calcula los N primeros primos*
- `int main (int argc, char **argv)`  
*Función principal del programa.*

### 4.2.1. Descripción detallada

Ejercicio para la comparación de tiempo: hilos.

Este fichero, en conjunto con el [ejercicio12b.c](#), sirven para comparar la eficiencia de los threads comparados con los procesos

#### Autor

Miguel Arconada Manteca y José Manuel Chacón Aguilera

#### Fecha

8-3-2018

### 4.2.2. Documentación de los 'defines'

#### 4.2.2.1. NUMERO\_HILOS

```
#define NUMERO_HILOS 100
```

Numero total de hilos que queremos crear

### 4.2.3. Documentación de los 'typedefs'



#### 4.2.3.1. estructura

```
typedef struct _estructura estructura
```

estructura generada al principio de la ejecución

Esta estructura almacena una cadena de 100 caracteres y un entero. Se usa para comparar el uso de memoria entre los hilos y los procesos

#### 4.2.4. Documentación de las funciones

##### 4.2.4.1. calcula\_primos()

```
void * calcula_primos (  
    void * arg )
```

calcula los N primeros primos

calcula\_primos calcula los N primeros números primos

##### Parámetros

<i>arg</i>	puntero a un entero que contiene N hecho casting a void*
------------	--

##### Devuelve

void

##### 4.2.4.2. main()

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

Función principal del programa.

Este programa lanza 100 hilos simultáneamente, en los que se calculan los N primeros primos

##### Parámetros

<i>argc</i>	numero de parametros de entrada
<i>argv</i>	array de cada parametro. El primero es el nombre del programa, y el segundo es N

**Devuelve**

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

### 4.3. Referencia del Archivo ejercicio13.c

Ejercicio sobre paso de parámetros en funciones.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <pthread.h>
#include <sys/types.h>
#include <sys/wait.h>
```

**Clases**

- struct [param](#)  
*Estructura de parámetros de un hilo.*

**defines**

- #define [BUFFER\\_SIZE](#) 256
- #define [ARRAY\\_SIZE](#) 30

**typedefs**

- typedef struct [param](#) [Param](#)  
*Estructura de parámetros de un hilo.*

**Funciones**

- void \* [calcula\\_matriz](#) (void \*arg)  
*multiplica una matriz por un escalar*
- int [main](#) ()  
*Función principal del programa.*

#### 4.3.1. Descripción detallada

Ejercicio sobre paso de parámetros en funciones.

Este fichero contiene nuestra solución al ejercicio 13 enunciado en el pdf de la práctica

**Autor**

Miguel Arconada Manteca y José Manuel Chacón Aguilera

**Fecha**

8-3-2018

### 4.3.2. Documentación de los 'defines'

#### 4.3.2.1. ARRAY\_SIZE

```
#define ARRAY_SIZE 30
```

Tamaño maximo de los buffers valores

#### 4.3.2.2. BUFFER\_SIZE

```
#define BUFFER_SIZE 256
```

Tamaño maximo del buffer cadena\_aux

### 4.3.3. Documentación de los 'typedefs'

#### 4.3.3.1. Param

```
typedef struct param Param
```

Estructura de parámetros de un hilo.

Esta estructura almacena la información necesaria para los hilos en este ejercicio

### 4.3.4. Documentación de las funciones

#### 4.3.4.1. calcula\_matriz()

```
void * calcula_matriz (  
    void * arg )
```

multiplica una matriz por un escalar

calcula\_matriz multiplica una matriz por un escalar. Está almacenada en forma de puntero a función para ser ejecutada por varios threads simultáneamente.

#### Parámetros

<i>arg</i>	estructura del tipo Param que contiene la matriz, el escalar, la dimension y un identificador del hilo
------------	--

**Devuelve**

void

**4.3.4.2. main()**

```
int main (
    void )
```

Función principal del programa.

Este programa pide por consola una dimensión, dos multiplicadores y dos matrices, y lanza dos hilos que multiplican cada matriz por los escalares, y que van imprimiendo cada línea según la calculan

**Parámetros**

void	
------	--

**Devuelve**

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

**4.4. Referencia del Archivo ejercicio4a.c**

Ejercicio 4a de la Práctica.

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <errno.h>
```

**defines**

- #define NUM\_PROC 6

**Funciones**

- int main (void)

**4.4.1. Descripción detallada**

Ejercicio 4a de la Práctica.

En este ejercicio vemos el ejemplo de ejecución de un programa que lanza procesos hijos.

**Autor**

José Manuel Chacón Aguilera y Miguel Arconada Manteca

**Fecha**

8-3-2018

#### 4.4.2. Documentación de los 'defines'

##### 4.4.2.1. NUM\_PROC

```
#define NUM_PROC 6
```

Número de iteraciones del bucle creador de procesos

### 4.5. Referencia del Archivo ejercicio4b.c

Ejercicio 4b de la Práctica.

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

#### defines

- #define NUM\_PROC 6

#### Funciones

- int main (void)

##### 4.5.1. Descripción detallada

Ejercicio 4b de la Práctica.

En este ejercicio vemos el ejemplo de ejecución de un programa que lanza procesos hijos.

#### Autor

José Manuel Chacón Aguilera y Miguel Arconada Manteca

#### Fecha

8-3-2018

##### 4.5.2. Documentación de los 'defines'

#### 4.5.2.1. NUM\_PROC

```
#define NUM_PROC 6
```

Número de iteraciones del bucle creador de procesos

### 4.6. Referencia del Archivo ejercicio5a.c

Ejercicio 5a de la Práctica.

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

#### defines

- #define NUM\_PROC 6

#### Funciones

- int main (void)

#### 4.6.1. Descripción detallada

Ejercicio 5a de la Práctica.

En este ejercicio vemos el ejemplo de ejecución de un programa que lanza procesos hijos.

#### Autor

José Manuel Chacón Aguilera y Miguel Arconada Manteca

#### Fecha

8-3-2018

#### 4.6.2. Documentación de los 'defines'

#### 4.6.2.1. NUM\_PROC

```
#define NUM_PROC 6
```

Número de iteraciones del bucle creador de procesos

## 4.7. Referencia del Archivo ejercicio5b.c

Ejercicio 5b de la Práctica.

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

### defines

- #define NUM\_PROC 6

### Funciones

- int **main** (void)

#### 4.7.1. Descripción detallada

Ejercicio 5b de la Práctica.

En este ejercicio vemos el ejemplo de ejecución de un programa que lanza procesos hijos.

#### Autor

José Manuel Chacón Aguilera y Miguel Arconada Manteca

#### Fecha

8-3-2018

#### 4.7.2. Documentación de los 'defines'

##### 4.7.2.1. NUM\_PROC

```
#define NUM_PROC 6
```

Número de iteraciones del bucle creador de procesos

## 4.8. Referencia del Archivo ejercicio6.c

Ejercicio 6 de la Práctica.

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

### defines

- `#define MAX_CAD 128`

### Funciones

- `int main ()`

#### 4.8.1. Descripción detallada

Ejercicio 6 de la Práctica.

En este ejercicio combinamos cuestiones teóricas sobre reserva de memoria dinamica y ejecución del programa en procesos distintos

#### Autor

José Manuel Chacón Aguilera y Miguel Arconada Manteca

#### Fecha

8-3-2018

#### 4.8.2. Documentación de los 'defines'

##### 4.8.2.1. MAX\_CAD

```
#define MAX_CAD 128
```

Tamaño máximo de las cadenas inicializadas



## 4.9. Referencia del Archivo ejercicio8.c

Ejercicio 8 de la Práctica.

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

### defines

- #define `MAX_PATH` 256

### Funciones

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

#### 4.9.1. Descripción detallada

Ejercicio 8 de la Práctica.

En este ejercicio lanzamos tantos procesos hijo como especificados en los parámetros de entrada de la función main y además hacemos que estos procesos hijos ejecuten un código distinto al de este programa todo esto lo conseguimos con una combinación de las rutinas fork() y la familia de rutinas exec().

### Autor

José Manuel Chacón Aguilera y Miguel Arconada Manteca

### Fecha

8-3-2018

#### 4.9.2. Documentación de los 'defines'

##### 4.9.2.1. MAX\_PATH

```
#define MAX_PATH 256
```

Tamaño máximo de la ruta

## 4.10. Referencia del Archivo ejercicio9.c

Ejercicio para la comunicacion entre procesos mediante tuberias.

```
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
#include <string.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <math.h>
```

### defines

- #define `LECTURA` 0
- #define `ESCRITURA` 1
- #define `BUFFER_SIZE` 512
- #define `MENSAJE_SIZE` 512

### Funciones

- int `factorial` (int n)  
*calcula el factorial de un numero*
- int `main` (int argc, char \*\*argv)  
*Función principal del programa.*

#### 4.10.1. Descripción detallada

Ejercicio para la comunicacion entre procesos mediante tuberias.

En este fichero hacemos uso de las pipes para implementar la comunicacion entre un proceso padre y cuatro procesos hijos, a los cuales les manda dos enteros, y estos devuelven el resultado de una operacion con ellos

#### Autor

Miguel Arconada Manteca y José Manuel Chacón Aguilera

#### Fecha

8-3-2018

#### 4.10.2. Documentación de los 'defines'

#### 4.10.2.1. BUFFER\_SIZE

```
#define BUFFER_SIZE 512
```

Tamaño máximo de los buffers

#### 4.10.2.2. ESCRITURA

```
#define ESCRITURA 1
```

Macro que define la zona de escritura de la pipe

#### 4.10.2.3. LECTURA

```
#define LECTURA 0
```

Macro que define la zona de lectura de la pipe

#### 4.10.2.4. MENSAJE\_SIZE

```
#define MENSAJE_SIZE 512
```

Tamaño máximo de los mensajes

### 4.10.3. Documentación de las funciones

#### 4.10.3.1. factorial()

```
int factorial (  
    int n )
```

calcula el factorial de un número

factorial calcula el factorial de un número pasado como argumento. Se calcula de forma recursiva, devolviendo el número multiplicado por el factorial del anterior, y con la condición de parada de que factorial(1) = 1

##### Parámetros

<i>arg</i>	estructura del tipo Param que contiene la matriz, el escalar, la dimensión y un identificador del hilo
------------	--

##### Devuelve

void

#### 4.10.3.2. main()

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

Función principal del programa.

Este programa crea cuatro procesos hijos que se comunican con el padre, e intercambian mensajes con numeros y el resultado de una operacion entre ellos diferente para cada hijo

##### Parámetros

<i>argc</i>	y <i>argv</i>
-------------	---------------

##### Devuelve

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

# Índice alfabético

`_estructura`, 5  
    `cadena`, 5  
    `entero`, 5

`ARRAY_SIZE`  
    `ejercicio13.c`, 13

`BUFFER_SIZE`  
    `ejercicio13.c`, 13  
    `ejercicio9.c`, 20

`cadena`  
    `_estructura`, 5

`calcula_matriz`  
    `ejercicio13.c`, 13

`calcula_primos`  
    `ejercicio12a.c`, 8  
    `ejercicio12b.c`, 11

`dim`  
    `param`, 6

`ESCRITURA`  
    `ejercicio9.c`, 21

`ejercicio12a.c`, 7  
    `calcula_primos`, 8  
    `estructura`, 8  
    `main`, 9  
    `NUMERO_PROCESOS`, 8

`ejercicio12b.c`, 9  
    `calcula_primos`, 11  
    `estructura`, 10  
    `main`, 11  
    `NUMERO_HILOS`, 10

`ejercicio13.c`, 12  
    `ARRAY_SIZE`, 13  
    `BUFFER_SIZE`, 13  
    `calcula_matriz`, 13  
    `main`, 14  
    `Param`, 13

`ejercicio4a.c`, 14  
    `NUM_PROC`, 15

`ejercicio4b.c`, 15  
    `NUM_PROC`, 15

`ejercicio5a.c`, 16  
    `NUM_PROC`, 16

`ejercicio5b.c`, 17  
    `NUM_PROC`, 17

`ejercicio6.c`, 18  
    `MAX_CAD`, 18

`ejercicio8.c`, 19  
    `MAX_PATH`, 19

`ejercicio9.c`, 20  
    `BUFFER_SIZE`, 20  
    `ESCRITURA`, 21  
    `factorial`, 21  
    `LECTURA`, 21  
    `MENSAJE_SIZE`, 21  
    `main`, 21

`entero`  
    `_estructura`, 5

`estructura`  
    `ejercicio12a.c`, 8  
    `ejercicio12b.c`, 10

`factorial`  
    `ejercicio9.c`, 21

`id`  
    `param`, 6

`LECTURA`  
    `ejercicio9.c`, 21

`MAX_CAD`  
    `ejercicio6.c`, 18

`MAX_PATH`  
    `ejercicio8.c`, 19

`MENSAJE_SIZE`  
    `ejercicio9.c`, 21

`main`  
    `ejercicio12a.c`, 9  
    `ejercicio12b.c`, 11  
    `ejercicio13.c`, 14  
    `ejercicio9.c`, 21

`matriz`  
    `param`, 6

`mult`  
    `param`, 6

`NUM_PROC`  
    `ejercicio4a.c`, 15  
    `ejercicio4b.c`, 15  
    `ejercicio5a.c`, 16  
    `ejercicio5b.c`, 17

`NUMERO_HILOS`  
    `ejercicio12b.c`, 10

`NUMERO_PROCESOS`  
    `ejercicio12a.c`, 8

`Param`

ejercicio13.c, [13](#)  
param, [6](#)  
dim, [6](#)  
id, [6](#)  
matriz, [6](#)  
mult, [6](#)