

# My Project

Generated by Doxygen 1.8.7

Fri Mar 2 2018 17:47:57



# 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	Argum Struct Reference . . . . .	5
3.1.1	Detailed Description . . . . .	5
3.1.2	Member Data Documentation . . . . .	5
3.1.2.1	dim . . . . .	5
3.1.2.2	id . . . . .	5
3.1.2.3	matrix . . . . .	5
3.1.2.4	pos1 . . . . .	5
3.1.2.5	pos2 . . . . .	6
3.1.2.6	scalar . . . . .	6
3.2	Structure Struct Reference . . . . .	6
3.2.1	Detailed Description . . . . .	6
3.2.2	Member Data Documentation . . . . .	6
3.2.2.1	n . . . . .	6
3.2.2.2	str . . . . .	6
<b>4</b>	<b>File Documentation</b>	<b>7</b>
4.1	ejercicio12a.c File Reference . . . . .	7
4.1.1	Detailed Description . . . . .	8
4.1.2	Macro Definition Documentation . . . . .	8
4.1.2.1	LEN . . . . .	8
4.1.2.2	N_CHILDS . . . . .	8
4.1.2.3	TENTOTHENINE . . . . .	8
4.1.3	Function Documentation . . . . .	8
4.1.3.1	calculate_primes . . . . .	8
4.1.3.2	is_prime . . . . .	8

4.2	<a href="#">ejercicio12b.c File Reference</a>	9
4.2.1	<a href="#">Detailed Description</a>	10
4.2.2	<a href="#">Macro Definition Documentation</a>	10
4.2.2.1	<a href="#">LEN</a>	10
4.2.2.2	<a href="#">N_CHILDS</a>	10
4.2.2.3	<a href="#">TENTOTHENINE</a>	10
4.2.3	<a href="#">Function Documentation</a>	10
4.2.3.1	<a href="#">calculate_primes</a>	10
4.2.3.2	<a href="#">is_prime</a>	10
4.3	<a href="#">ejercicio13.c File Reference</a>	11
4.3.1	<a href="#">Detailed Description</a>	11
4.3.2	<a href="#">Macro Definition Documentation</a>	12
4.3.2.1	<a href="#">LEN</a>	12
4.3.3	<a href="#">Function Documentation</a>	12
4.3.3.1	<a href="#">mult_matrix</a>	12
4.4	<a href="#">ejercicio4a.c File Reference</a>	12
4.4.1	<a href="#">Detailed Description</a>	13
4.4.2	<a href="#">Macro Definition Documentation</a>	13
4.4.2.1	<a href="#">PROC_NUM</a>	13
4.5	<a href="#">ejercicio4b.c File Reference</a>	13
4.5.1	<a href="#">Detailed Description</a>	14
4.5.2	<a href="#">Macro Definition Documentation</a>	14
4.5.2.1	<a href="#">PROC_NUM</a>	14
4.6	<a href="#">ejercicio5a.c File Reference</a>	14
4.6.1	<a href="#">Detailed Description</a>	15
4.6.2	<a href="#">Macro Definition Documentation</a>	15
4.6.2.1	<a href="#">PROC_NUM</a>	15
4.7	<a href="#">ejercicio5b.c File Reference</a>	15
4.7.1	<a href="#">Detailed Description</a>	16
4.7.2	<a href="#">Macro Definition Documentation</a>	16
4.7.2.1	<a href="#">PROC_NUM</a>	16
4.8	<a href="#">ejercicio6.c File Reference</a>	16
4.8.1	<a href="#">Detailed Description</a>	17
4.8.2	<a href="#">Macro Definition Documentation</a>	17
4.8.2.1	<a href="#">LEN</a>	17
4.9	<a href="#">ejercicio8_1.c File Reference</a>	17
4.9.1	<a href="#">Detailed Description</a>	18
4.9.2	<a href="#">Macro Definition Documentation</a>	18
4.9.2.1	<a href="#">PATH_LEN</a>	18
4.10	<a href="#">ejercicio8_2.c File Reference</a>	18

---

4.10.1 Detailed Description . . . . .	19
4.10.2 Macro Definition Documentation . . . . .	19
4.10.2.1 PATH_LEN . . . . .	19
4.11 ejercicio9.c File Reference . . . . .	19
4.11.1 Detailed Description . . . . .	20
4.11.2 Macro Definition Documentation . . . . .	20
4.11.2.1 LEN . . . . .	20
4.11.2.2 N_CHILDS . . . . .	21
4.11.2.3 READ . . . . .	21
4.11.2.4 WRITE . . . . .	21
4.11.3 Function Documentation . . . . .	21
4.11.3.1 abs . . . . .	21
4.11.3.2 factorial . . . . .	21
4.11.3.3 split_first . . . . .	21
4.11.3.4 split_second . . . . .	21
<b>Index</b>	<b>23</b>



# Chapter 1

## Class Index

### 1.1 Class List

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

<a href="#">Argum</a>	Estructura de argumentos de entrada . . . . .	5
<a href="#">Structure</a>	Estructura programa . . . . .	6





## Chapter 2

# File Index

### 2.1 File List

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

<a href="#">ejercicio12a.c</a>	Ejercicio12a Creacion de 100 procesos y en cada uno se calculan los primeros N numeros primos, siendo N pasado como argumento entrada al programa . . . . .	7
<a href="#">ejercicio12b.c</a>	Ejercicio12b Creacion de 100 hilos y en cada uno se calculan los primeros N numeros primos, siendo N pasado como argumento entrada al programa . . . . .	9
<a href="#">ejercicio13.c</a>	Ejercicio13 Creacion de dos hilos. Cada uno multiplica una matriz por un escalar, imprimiendo el resultado cada vez que acaba de multiplicar una fila. Ademas, cada hilo sabe por donde va el otro . . . . .	11
<a href="#">ejercicio4a.c</a>	Ejercicio4a Modificación del código dado para que cada HIJO imprima su PID y el PID de su padre . . . . .	12
<a href="#">ejercicio4b.c</a>	Ejercicio4b Modificación del código dado para que cada HIJO imprima su PID y el PID de su padre. Se diferencia con el ejercicio anterior por la presencia de wait() . . . . .	13
<a href="#">ejercicio5a.c</a>	Ejercicio5a Modificación del ejercicio4b para que cada proceso tengo un único hijo que sea esperado por su padre . . . . .	14
<a href="#">ejercicio5b.c</a>	Ejercicio5b Modificación del ejercicio4b para que un proceso tenga un conjunto de hijos que serán esperados por el padre . . . . .	15
<a href="#">ejercicio6.c</a>	Ejercicio6 Comprobación de si dos procesos (padre e hijo) comparten la misma zona de memoria una vez lanzados . . . . .	16
<a href="#">ejercicio8_1.c</a>	Ejercicio8_1 Ejecución de varios programas desde el primero pasado como argumento de entrada hasta el ultimo . . . . .	17
<a href="#">ejercicio8_2.c</a>	Ejercicio8_2 Ejecución de varios programas desde el ultimo pasado como argumento de entrada hasta el primero . . . . .	18
<a href="#">ejercicio9.c</a>	Ejercicio9 Ejercicio de comunicación bidireccional entre procesos utilizando tuberías (pipes) .	19



## Chapter 3

# Class Documentation

### 3.1 Argum Struct Reference

Estructura de argumentos de entrada.

#### Public Attributes

- int [id](#)
- int \*\* [matrix](#)
- int [dim](#)
- int [scalar](#)
- char \* [pos1](#)
- char \* [pos2](#)

#### 3.1.1 Detailed Description

Estructura de argumentos de entrada.

La mision de esta enctructura es pasarle todos los argumentos de entrada que necesitan los hilos para su ejecucion

#### 3.1.2 Member Data Documentation

##### 3.1.2.1 int Argum::dim

dimension de la matriz

##### 3.1.2.2 int Argum::id

Id del hilo

##### 3.1.2.3 int\*\* Argum::matrix

matriz de enteros

##### 3.1.2.4 char\* Argum::pos1

variable donde hilo1 lee e hilo2 escribe

### 3.1.2.5 char\* Argum::pos2

variable donde hilo1 escribe e hilo2 lee

### 3.1.2.6 int Argum::scalar

escalar que multiplica la matriz

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

- [ejercicio13.c](#)

## 3.2 Structure Struct Reference

Estructura programa.

### Public Attributes

- char [str](#) [[LEN](#)]
- int [n](#)

### 3.2.1 Detailed Description

Estructura programa.

Esta estructura contiene una cadena de caracteres y un entero.

### 3.2.2 Member Data Documentation

#### 3.2.2.1 int Structure::n

numero entero

#### 3.2.2.2 char Structure::str

cadena de caracteres

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

- [ejercicio12a.c](#)
- [ejercicio12b.c](#)
- [ejercicio6.c](#)

## Chapter 4

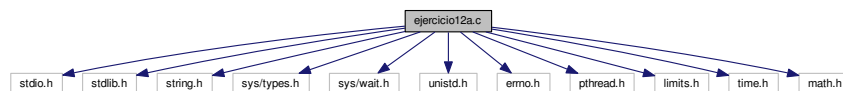
# File Documentation

### 4.1 ejercicio12a.c File Reference

Ejercicio12a Creacion de 100 procesos y en cada uno se calculan los primeros N numeros primos, siendo N pasado como argumento entrada al programa.

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

Include dependency graph for ejercicio12a.c:



### Classes

- struct [Structure](#)  
*Estructura programa.*

### Macros

- #define [LEN](#) 100
- #define [TENTOTHENINE](#) 1000000000
- #define [N\\_CHILDS](#) 100

### Functions

- int [is\\_prime](#) (int n)

*evalua si un numero es primo o no.*

- int \* `calculate_primes` (int `n_primes`)  
*devuelve un array con los `n_primes` primeros primos*
- int **main** (int argc, char const \*argv[])

#### 4.1.1 Detailed Description

Ejercicio12a Creacion de 100 procesos y en cada uno se calculan los primeros N numeros primos, siendo N pasado como argumento entrada al programa.

##### Author

Alejandro Santorum & David Cabornero

##### Version

1.0

##### Date

02-03-2018

#### 4.1.2 Macro Definition Documentation

##### 4.1.2.1 #define LEN 100

Longitud de las cadenas de caracteres

##### 4.1.2.2 #define N\_CHILDS 100

Numero de procesos hijo

##### 4.1.2.3 #define TENTOTHENINE 1000000000

Constante

#### 4.1.3 Function Documentation

##### 4.1.3.1 int \* `calculate_primes` ( int `n_primes` )

devuelve un array con los `n_primes` primeros primos

##### Parameters

<code>n_primes</code>	numero de primeros primos a ser calculados
-----------------------	--------------------------------------------

##### Returns

array de primos

##### 4.1.3.2 int `is_prime` ( int `n` )

evalua si un numero es primo o no.

## Parameters

$n$	entero a ser evaluado
-----	-----------------------

## Returns

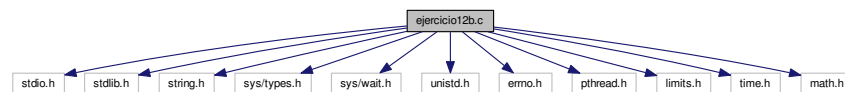
true si es primo, false si no.

## 4.2 ejercicio12b.c File Reference

Ejercicio12b Creacion de 100 hilos y en cada uno se calculan los primeros N numeros primos, siendo N pasado como argumento entrada al programa.

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

Include dependency graph for ejercicio12b.c:



## Classes

- struct [Structure](#)  
*Estructura programa.*

## Macros

- #define [LEN](#) 100
- #define [TENTOTHENINE](#) 1000000000
- #define [N\\_CHILDS](#) 100

## Functions

- int [is\\_prime](#) (int n)  
*evalua si un numero es primo o no.*
- void \* [calculate\\_primes](#) (void \*arg)  
*devuelve un array con los n\_primes primeros primos*
- int **main** (int argc, char const \*argv[])

### 4.2.1 Detailed Description

Ejercicio12b Creacion de 100 hilos y en cada uno se calculan los primeros N numeros primos, siendo N pasado como argumento entrada al programa.

#### Author

Alejandro Santorum & David Cabornero

#### Version

1.0

#### Date

02-03-2018

### 4.2.2 Macro Definition Documentation

#### 4.2.2.1 #define LEN 100

Longitud de las cadenas de caracteres

#### 4.2.2.2 #define N\_CHILDS 100

Numero de procesos hijo

#### 4.2.2.3 #define TENTOTHENINE 1000000000

Constante

### 4.2.3 Function Documentation

#### 4.2.3.1 void \* calculate\_primes ( void \* arg )

devuelve un array con los n\_primes primeros primos

#### Parameters

<i>arg,estructura</i>	con todos los argumentos de la fuincion que queramos
-----------------------	------------------------------------------------------

#### Returns

void\*, debido a la utilizacion de hilos (threads)

#### 4.2.3.2 int is\_prime ( int n )

evalua si un numero es primo o no.

#### Parameters



<i>n</i>	entero a ser evaluado
----------	-----------------------

## Returns

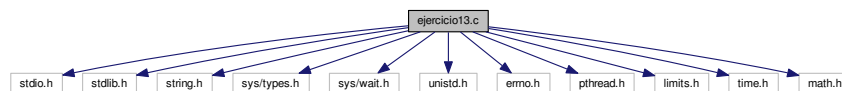
true si es primo, false si no.

## 4.3 ejercicio13.c File Reference

Ejercicio13 Creacion de dos hilos. Cada uno multiplica una matriz por un escalar, imprimiendo el resultado cada vez que acaba de multiplicar una fila. Ademàs, cada hilo sabe por donde va el otro.

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

Include dependency graph for ejercicio13.c:



## Classes

- struct [Argum](#)  
*Estructura de argumentos de entrada.*

## Macros

- #define [LEN](#) 50

## Functions

- void \* [mult\\_matrix](#) (void \*arg)  
*multiplica una matriz por un escalar. Ademàs, indica el estado del hilo hermano.*
- int **main** (int argc, char const \*argv[])

### 4.3.1 Detailed Description

Ejercicio13 Creacion de dos hilos. Cada uno multiplica una matriz por un escalar, imprimiendo el resultado cada vez que acaba de multiplicar una fila. Ademàs, cada hilo sabe por donde va el otro.

**Author**

Alejandro Santorum &amp; David Cabornero

**Version**

1.0

**Date**

02-03-2018

**4.3.2 Macro Definition Documentation****4.3.2.1 #define LEN 50**

Longitud cadena de caracteres

**4.3.3 Function Documentation****4.3.3.1 void \* mult\_matrix ( void \* arg )**

multiplica una matrix por un escalar. Ademas, indica el estado del hilo hermano.

**Parameters**

<i>*arg</i>	estructura con todos los parametros necesarios
-------------	------------------------------------------------

**Returns**

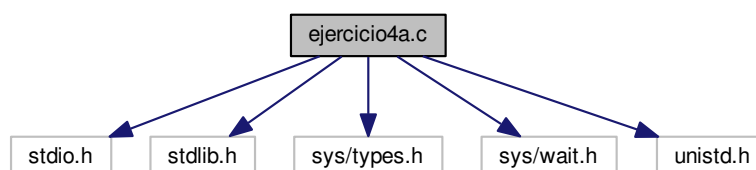
void\*, debido a politica de hilos

**4.4 ejercicio4a.c File Reference**

Ejercicio4a Modificación del código dado para que cada HIJO imprima su PID y el PID de su padre.

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

Include dependency graph for ejercicio4a.c:



## Macros

- `#define PROC_NUM 6`

## Functions

- `int main (void)`

### 4.4.1 Detailed Description

Ejercicio4a Modificación del código dado para que cada HIJO imprima su PID y el PID de su padre.

#### Author

Alejandro Santorum & David Cabornero

#### Version

1.0

#### Date

02-03-2018

### 4.4.2 Macro Definition Documentation

#### 4.4.2.1 `#define PROC_NUM 6`

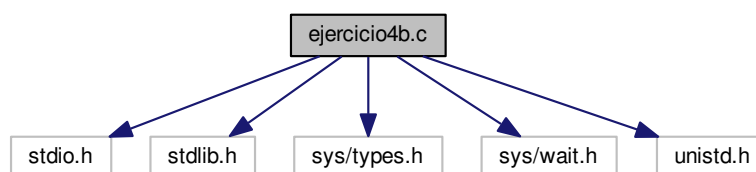
Número de procesos

## 4.5 ejercicio4b.c File Reference

Ejercicio4b Modificación del código dado para que cada HIJO imprima su PID y el PID de su padre. Se diferencia con el ejercicio anterior por la presencia de `wait()`.

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

Include dependency graph for ejercicio4b.c:



## Macros

- `#define PROC_NUM 6`

## Functions

- `int main (void)`

### 4.5.1 Detailed Description

Ejercicio4b Modificación del código dado para que cada HIJO imprima su PID y el PID de su padre. Se diferencia con el ejercicio anterior por la presencia de `wait()`.

#### Author

Alejandro Santorum & David Cabornero

#### Version

1.0

#### Date

02-03-2018

### 4.5.2 Macro Definition Documentation

#### 4.5.2.1 `#define PROC_NUM 6`

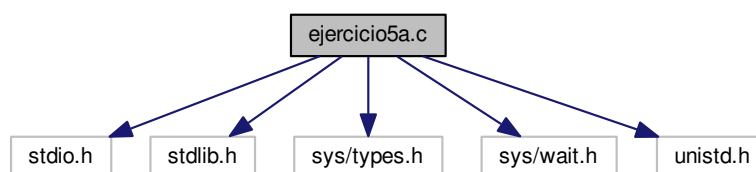
Número de procesos

## 4.6 ejercicio5a.c File Reference

Ejercicio5a Modificación del ejercicio4b para que cada proceso tengo un único hijo que sea esperado por su padre.

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

Include dependency graph for ejercicio5a.c:



## Macros

- `#define PROC_NUM 6`

## Functions

- `int main (void)`

### 4.6.1 Detailed Description

Ejercicio5a Modificación del ejercicio4b para que cada proceso tengo un único hijo que sea esperado por su padre.

#### Author

Alejandro Santorum & David Cabornero

#### Version

1.0

#### Date

02-03-2018

### 4.6.2 Macro Definition Documentation

#### 4.6.2.1 `#define PROC_NUM 6`

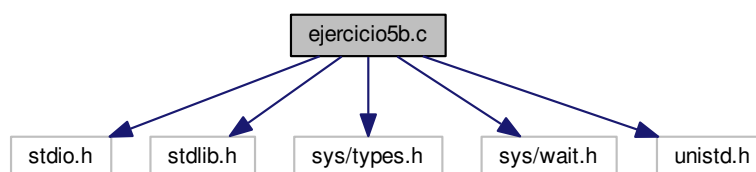
Número de procesos

## 4.7 ejercicio5b.c File Reference

Ejercicio5b Modificación del ejercicio4b para que un proceso tenga un conjunto de hijos que serán esperados por el padre.

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

Include dependency graph for ejercicio5b.c:



## Macros

- `#define PROC_NUM 6`

## Functions

- `int main (void)`

### 4.7.1 Detailed Description

Ejercicio5b Modificación del ejercicio4b para que un proceso tenga un conjunto de hijos que serán esperados por el padre.

#### Author

Alejandro Santorum & David Cabornero

#### Version

1.0

#### Date

02-03-2018

### 4.7.2 Macro Definition Documentation

#### 4.7.2.1 `#define PROC_NUM 6`

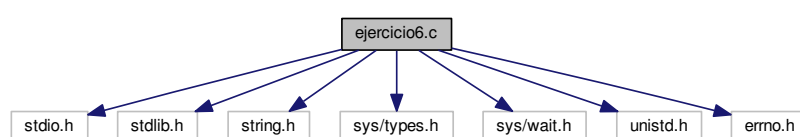
Número de procesos

## 4.8 ejercicio6.c File Reference

Ejercicio6 Comprobación de si dos procesos (padre e hijo) comparten la misma zona de memoria una vez lanzados.

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

Include dependency graph for ejercicio6.c:



## Classes

- struct [Structure](#)

*Estructura programa.*

## Macros

- #define [LEN](#) 80

## Functions

- int **main** ()

### 4.8.1 Detailed Description

Ejercicio6 Comprobación de si dos procesos (padre e hijo) comparten la misma zona de memoria una vez lanzados.

#### Author

Alejandro Santorum & David Cabornero

#### Version

1.0

#### Date

02-03-2018

### 4.8.2 Macro Definition Documentation

#### 4.8.2.1 #define LEN 80

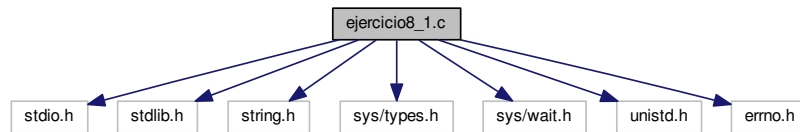
Longitud cadena de caracteres

## 4.9 ejercicio8\_1.c File Reference

Ejercicio8\_1 Ejecución de varios programas desde el primero pasado como argumento de entrada hasta el ultimo.

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

Include dependency graph for ejercicio8\_1.c:



## Macros

- `#define PATH_LEN 100`

## Functions

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

### 4.9.1 Detailed Description

Ejercicio8\_1 Ejecución de varios programas desde el primero pasado como argumento de entrada hasta el ultimo.

#### Author

Alejandro Santorum & David Cabornero

#### Version

1.0

#### Date

02-03-2018

### 4.9.2 Macro Definition Documentation

#### 4.9.2.1 `#define PATH_LEN 100`

Longitud cadenas de caracteres

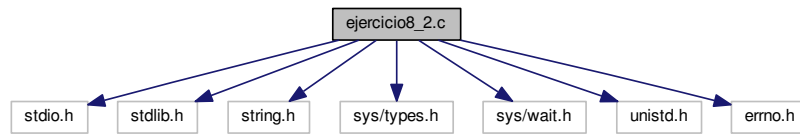
## 4.10 ejercicio8\_2.c File Reference

Ejercicio8\_2 Ejecución de varios programas desde el ultimo pasado como argumento de entrada hasta el primero.

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



Include dependency graph for ejercicio8\_2.c:



## Macros

- `#define PATH_LEN 100`

## Functions

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

### 4.10.1 Detailed Description

Ejercicio8\_2 Ejecución de varios programas desde el ultimo pasado como argumento de entrada hasta el primero.

#### Author

Alejandro Santorum & David Cabornero

#### Version

1.0

#### Date

02-03-2018

### 4.10.2 Macro Definition Documentation

#### 4.10.2.1 `#define PATH_LEN 100`

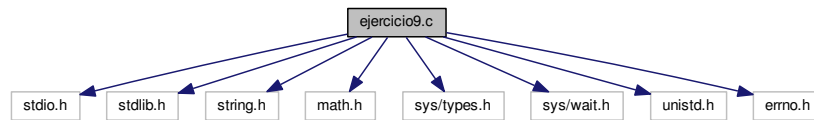
Longitud cadenas de caracteres

## 4.11 ejercicio9.c File Reference

Ejercicio9 Ejercicio de comunicación bidireccional entre procesos utilizando tuberías (pipes)

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

Include dependency graph for ejercicio9.c:



## Macros

- `#define READ 0`
- `#define WRITE 1`
- `#define N_CHILDs 4`
- `#define LEN 200`

## Functions

- `int split_first (char *str)`  
*separa de una cadena el primero Operando*
- `int split_second (char *str)`  
*separa de una cadena el segundo Operando*
- `int factorial (int a)`  
*calcula el factorial*
- `int abs (int a)`  
*calcula el valor absoluto*
- `int main ()`

### 4.11.1 Detailed Description

Ejercicio9 Ejercicio de comunicación bidireccional entre procesos utilizando tuberías (pipes)

#### Author

Alejandro Santorum & David Cabornero

#### Version

1.0

#### Date

02-03-2018

### 4.11.2 Macro Definition Documentation

#### 4.11.2.1 `#define LEN 200`

Longitud de las cadenas de caracteres

4.11.2.2 `#define N_CHILDS 4`

Numero de procesos hijo

4.11.2.3 `#define READ 0`

Macro para lectura en tuberías

4.11.2.4 `#define WRITE 1`

Macro para escritura en tuberías

## 4.11.3 Function Documentation

4.11.3.1 `int abs ( int a )`

calcula el valor absoluto

## Parameters

<i>a</i>	entero para calcular su valor absoluto
----------	----------------------------------------

## Returns

entero valor absoluto

4.11.3.2 `int factorial ( int a )`

calcula el factorial

## Parameters

<i>a</i>	entero para calcular su factorial
----------	-----------------------------------

## Returns

entero factorial

4.11.3.3 `int split_first ( char * str )`

separa de una cadena el primero Operando

## Parameters

<i>str</i>	cadena para ser separada
------------	--------------------------

## Returns

entero que es el primer operando

4.11.3.4 `int split_second ( char * str )`

separa de una cadena el segundo Operando

**Parameters**

<i>str</i>	cadena para ser separada
------------	--------------------------

**Returns**

entero que es el segundo operando

# Index

- Argum, [5](#)
  - dim, [5](#)
  - id, [5](#)
  - matrix, [5](#)
  - pos1, [5](#)
  - pos2, [5](#)
  - scalar, [6](#)
- dim
  - Argum, [5](#)
- id
  - Argum, [5](#)
- matrix
  - Argum, [5](#)
- n
  - Structure, [6](#)
- pos1
  - Argum, [5](#)
- pos2
  - Argum, [5](#)
- scalar
  - Argum, [6](#)
- str
  - Structure, [6](#)
- Structure, [6](#)
  - n, [6](#)
  - str, [6](#)