

TP SSOO KernelCrafters

1

Generated by Doxygen 1.9.1

1 Data Structure Index	1
1.1 Data Structures	1
2 File Index	3
2.1 File List	3
3 Data Structure Documentation	5
3.1 t_buffer Struct Reference	5
3.1.1 Detailed Description	5
3.2 t_packet Struct Reference	5
3.2.1 Detailed Description	6
3.3 t_process_conection_args Struct Reference	6
3.3.1 Detailed Description	6
4 File Documentation	7
4.1 utils/include/comunication.h File Reference	7
4.2 utils/include/configs.h File Reference	7
4.2.1 Detailed Description	8
4.2.2 Function Documentation	8
4.2.2.1 end_program()	8
4.2.2.2 initialize_config()	8
4.3 utils/include/logger.h File Reference	9
4.3.1 Detailed Description	9
4.3.2 Function Documentation	10
4.3.2.1 initialize_logger()	10
4.4 utils/include/opcode.h File Reference	11
4.4.1 Detailed Description	11
4.5 utils/include/protocol.h File Reference	12
4.5.1 Detailed Description	13
4.5.2 Function Documentation	13
4.5.2.1 add_to_packet()	13
4.5.2.2 create_buffer()	13
4.5.2.3 create_packet()	14
4.5.2.4 destroy_packet()	14
4.5.2.5 fetch_buffer()	14
4.5.2.6 fetch_codop()	15
4.5.2.7 fetch_packet()	15
4.5.2.8 send_packet()	15
4.5.2.9 serialize_packet()	16
4.6 utils/include/sockets.h File Reference	16
4.6.1 Detailed Description	17
4.6.2 Function Documentation	17
4.6.2.1 close_conection()	17

4.6.2.2 create_conection()	17
4.6.2.3 initialize_server()	18
4.6.2.4 wait_client()	18
4.7 utils/include/utils.h File Reference	19
4.7.1 Detailed Description	19
Index	21

Chapter 1

Data Structure Index

1.1 Data Structures

Here are the data structures with brief descriptions:

t_buffer	Struct to handle the buffer of the packet	5
t_packet	Struct to handle the packet	5
t_process_conection_args	Struct to pass arguments to the process_conection function	6

Chapter 2

File Index

2.1 File List

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

entradasalida/include/ main.h	??
kernel/include/ main.h	??
memoria/include/ main.h	??
utils/include/ communication.h	
Handles the communication between the server and the client using Multithreading	7
utils/include/ configs.h	
Config functions to read the configuration files	7
utils/include/ logger.h	
Logger Functions to log messages	9
utils/include/ opcode.h	
Operation code enum to define the operation code of the packet sent so the make some specific actions	11
utils/include/ protocol.h	
Handles the protocol of the communication between the client and the server	12
utils/include/ sockets.h	
Socket Connection Functions	16
utils/include/ utils.h	
Include all the headers from the project	19

Chapter 3

Data Structure Documentation

3.1 t_buffer Struct Reference

Struct to handle the buffer of the packet.

```
#include <protocol.h>
```

Data Fields

- int **size**
- void * **stream**

3.1.1 Detailed Description

Struct to handle the buffer of the packet.

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

- [utils/include/protocol.h](#)

3.2 t_packet Struct Reference

struct to handle the packet

```
#include <protocol.h>
```

Collaboration diagram for t_packet:

Data Fields

- [op_code](#) **code**
- [t_buffer](#) * **buffer**

3.2.1 Detailed Description

struct to handle the packet

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

- [utils/include/protocol.h](#)

3.3 t_process_conection_args Struct Reference

Struct to pass arguments to the process_conection function.

```
#include <communication.h>
```

Data Fields

- `t_log * logger`
- `int fd`
- `char * server_name`

3.3.1 Detailed Description

Struct to pass arguments to the process_conection function.

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

- [utils/include/communication.h](#)

Chapter 4

File Documentation

4.1 utils/include/comunication.h File Reference

Handles the communication between the server and the client using Multithreading.

```
#include <stdio.h>
#include <stdlib.h>
#include <commons/log.h>
#include <inttypes.h>
#include <pthread.h>
#include "protocol.h"
#include "sockets.h"
#include "opcode.h"
```

Include dependency graph for comunication.h:

4.2 utils/include/configs.h File Reference

Config functions to read the configuration files.

```
#include <stdio.h>
#include <stdlib.h>
#include <commons/log.h>
#include <commons/string.h>
#include <commons/config.h>
#include "readline/readline.h"
```

Include dependency graph for configs.h: This graph shows which files directly or indirectly include this file:

Functions

- `t_config * initialize_config (t_log *logger, char *path)`
Initializes the config functions of the Commons library to read the configuration files.
- `void end_program (t_log *logger, t_config *config)`
Destroy config and logger.

4.2.1 Detailed Description

Config functions to read the configuration files.

Author

KernelCrafters

Version

1.0

Date

2024-04-18

Copyright

Copyright (c) 2024

4.2.2 Function Documentation

4.2.2.1 end_program()

```
void end_program (
    t_log * logger,
    t_config * config )
```

Destroy config and logger.

Parameters

<i>logger</i>	logger created by initialize_logger
<i>config</i>	config var created by initialize_config

4.2.2.2 initialize_config()

```
t_config* initialize_config (
    t_log * logger,
    char * path )
```

Initializes the config functions of the Commons library to read the configuration files.

Parameters

<i>logger</i>	logger from commons library
<i>path</i>	path to the configuration file

Returns

t_config*

4.3 utils/include/logger.h File Reference

Logger Functions to log messages.

```
#include <stdio.h>
#include <stdlib.h>
#include <commons/log.h>
#include <commons/string.h>
#include <commons/config.h>
#include <readline/readline.h>
```

Include dependency graph for logger.h: This graph shows which files directly or indirectly include this file:

Functions

- t_log * [initialize_logger](#) (char *logger_name, char *process_name, bool visible, t_log_level log_level)
Creates a logger from the commons library to log messages.

4.3.1 Detailed Description

Logger Functions to log messages.

Author

KernelCrafters

Version

1.0

Date

2024-04-18

Copyright

Copyright (c) 2024

4.3.2 Function Documentation

4.3.2.1 initialize_logger()

```
t_log* initialize_logger (
    char * logger_name,
    char * process_name,
    bool visible,
    t_log_level log_level )
```

Creates a logger from the commons library to log messages.

Parameters

<i>logger_name</i>	Name of the logger
<i>process_name</i>	Name of the process
<i>visible</i>	True if visible or false otherwise
<i>log_level</i>	log level defined in t_log_level

Returns

t_log*

4.4 utils/include/opcode.h File Reference

Operation code enum to define the operation code of the packet sent so the make some specific actions.

This graph shows which files directly or indirectly include this file:

Enumerations

- enum `op_code` {
 MENSAJE , **HANDSHAKE_KERNEL** , **HANDSHAKE_ENTRADA_SALIDA** , **HANDSHAKE_CPU** ,
 EJECUTAR_SCRIPT , **INICIAR_PROCESO** , **FINALIZAR_PROCESO** , **DETENER_PLANIFICACION** ,
 INICIAR_PLANIFICACION , **PROCESO_ESTADO** , **PACKET** }

Enum to define the operation code of the packet sent.

4.4.1 Detailed Description

Operation code enum to define the operation code of the packet sent so the make some specific actions.

Author

KernelCrafters

Version

1.0

Date

2024-04-18

Copyright

Copyright (c) 2024

4.5 utils/include/protocol.h File Reference

Handles the protocol of the communication between the client and the server.

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/socket.h>
#include <signal.h>
#include <unistd.h>
#include <netdb.h>
#include <commons/log.h>
#include <commons/collections/list.h>
#include <string.h>
#include <assert.h>
#include "opcode.h"
```

Include dependency graph for protocol.h: This graph shows which files directly or indirectly include this file:

Data Structures

- struct [t_buffer](#)
Struct to handle the buffer of the packet.
- struct [t_packet](#)
struct to handle the packet

Functions

- [t_buffer](#) * [create_buffer](#) ()
Create a buffer object.
- [t_packet](#) * [create_packet](#) ([op_code](#) code, [t_buffer](#) *buffer)
Create a packet object that contains the operation code and the buffer to send.
- void [add_to_packet](#) ([t_packet](#) *packet, void *stream, int size)
Add the stream to the packet.
- void * [serialize_packet](#) ([t_packet](#) *packet, int buffer_size)
Serializes the packet to send it.
- void [destroy_packet](#) ([t_packet](#) *packet)
Destroys the packet freeing the memory. It is used by send_packet function.
- void [send_packet](#) ([t_packet](#) *packet, int client_socket)
Sends the packet to the server.
- [t_list](#) * [fetch_packet](#) (int client_socket)
Fetch the packet from the client.
- void * [fetch_buffer](#) (int *size, int client_socket)
Fetch the buffer from the client. It is used by fetch_packet function.
- int [fetch_codop](#) (int client_socket)
Fetch the operation code from the client.

4.5.1 Detailed Description

Handles the protocol of the communication between the client and the server.

Author

your name (you@domain.com)

Version

1.0

Date

2024-04-18

Copyright

Copyright (c) 2024

4.5.2 Function Documentation

4.5.2.1 add_to_packet()

```
void add_to_packet (
    t_packet * packet,
    void * stream,
    int size )
```

Add the stream to the packet.

Parameters

<i>packet</i>	Packet to add the stream
<i>stream</i>	Data of the buffer
<i>size</i>	size of the stream

4.5.2.2 create_buffer()

```
t_buffer* create_buffer ( )
```

Create a buffer object.

Returns

`t_buffer*`

Implementar stream

4.5.2.3 create_packet()

```
t_packet* create_packet (
    op_code code,
    t_buffer * buffer )
```

Create a packet object that contains the operation code and the buffer to send.

Parameters

<i>code</i>	operation code
<i>buffer</i>	buffer to send

Returns

`t_packet*`

4.5.2.4 destroy_packet()

```
void destroy_packet (
    t_packet * packet )
```

Destroys the packet freeing the memory. It is used by send_packet function.

Parameters

<i>packet</i>	packet to destroy
---------------	-------------------

4.5.2.5 fetch_buffer()

```
void* fetch_buffer (
    int * size,
    int client_socket )
```

Fetch the buffer from the client. It is used by fetch_packet function.

Parameters

<i>size</i>	size of the buffer
<i>client_socket</i>	client file descriptor

Returns

void*

4.5.2.6 fetch_codop()

```
int fetch_codop (
    int client_socket )
```

Fetch the operation code from the client.

Parameters

<i>client_socket</i>	client file descriptor
----------------------	------------------------

Returns

int Returns operation code or -1 if there is an error

4.5.2.7 fetch_packet()

```
t_list* fetch_packet (
    int client_socket )
```

Fetch the packet from the client.

Parameters

<i>client_socket</i>	client file descriptor
----------------------	------------------------

Returns

t_list* List of packets

4.5.2.8 send_packet()

```
void send_packet (
    t_packet * packet,
    int client_socket )
```

Sends the packet to the server.

Parameters

<i>packet</i>	packet to send
<i>client_socket</i>	client file descriptor

4.5.2.9 serialize_packet()

```
void* serialize_packet (
    t_packet * packet,
    int buffer_size )
```

Serializes the packet to send it.

Parameters

<i>packet</i>	packet to serialize
<i>buffer_size</i>	size of the buffer

Returns

void*

4.6 utils/include/sockets.h File Reference

Socket Connection Functions.

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/socket.h>
#include <unistd.h>
#include <netdb.h>
#include <commons/log.h>
#include <commons/collections/list.h>
#include <string.h>
#include <assert.h>
#include <pthread.h>
```

Include dependency graph for sockets.h: This graph shows which files directly or indirectly include this file:

Functions

- int [create_conection](#) (t_log *logger, char *ip, char *port)
Create conection with server.
- void [close_conection](#) (int *client_socket)
Close conection with server.
- int [initialize_server](#) (t_log *logger, const char *name, char *ip, char *port)
initialize the server
- int [wait_client](#) (t_log *logger, const char *name, int server_socket)
Listen to the server. It is used by [server_listen\(\)](#)

4.6.1 Detailed Description

Socket Connection Functions.

Author

KernelCrafters

Version

1.0

Date

2024-04-16

Copyright

Copyright (c) 2024

4.6.2 Function Documentation

4.6.2.1 close_conection()

```
void close_conection (
    int * client_socket )
```

Close conection with server.

Parameters

<i>client_socket</i>	client_fd to close connection
----------------------	-------------------------------

4.6.2.2 create_conection()

```
int create_conection (
    t_log * logger,
    char * ip,
    char * port )
```

Create conection with server.

Parameters

<i>logger</i>	Logger from commons libraries
<i>ip</i>	ip to connect
<i>port</i>	Port to connect

Returns

int Return client_socket

4.6.2.3 initialize_server()

```
int initialize_server (
    t_log * logger,
    const char * name,
    char * ip,
    char * port )
```

initialize the server

Parameters

<i>logger</i>	logger from commons libraries
<i>name</i>	name of the server
<i>ip</i>	ip to connect
<i>port</i>	Ip port to connect

Returns

int return server_socket

4.6.2.4 wait_client()

```
int wait_client (
    t_log * logger,
    const char * name,
    int server_socket )
```

Listen to the server. It is used by [server_listen\(\)](#)

Parameters

<i>logger</i>	logger from commons libraries
<i>name</i>	name of the server
<i>server_socket</i>	server file descriptor to listen

Returns

int return client_socket

4.7 utils/include/utils.h File Reference

Include all the headers from the project.

```
#include <commons/log.h>
#include <commons/string.h>
#include <commons/config.h>
#include <readline/readline.h>
#include "sockets.h"
#include "logger.h"
#include "configs.h"
#include "protocol.h"
#include "opcode.h"
#include "communication.h"
Include dependency graph for utils.h:
```

4.7.1 Detailed Description

Include all the headers from the project.

Author

your name (you@domain.com)

Version

1.0

Date

2024-04-18

Copyright

Copyright (c) 2024

Index

- add_to_packet
 - protocol.h, [13](#)
- close_conection
 - sockets.h, [17](#)
- configs.h
 - end_program, [8](#)
 - initialize_config, [8](#)
- create_buffer
 - protocol.h, [13](#)
- create_conection
 - sockets.h, [17](#)
- create_packet
 - protocol.h, [14](#)
- destroy_packet
 - protocol.h, [14](#)
- end_program
 - configs.h, [8](#)
- fetch_buffer
 - protocol.h, [14](#)
- fetch_codop
 - protocol.h, [15](#)
- fetch_packet
 - protocol.h, [15](#)
- initialize_config
 - configs.h, [8](#)
- initialize_logger
 - logger.h, [10](#)
- initialize_server
 - sockets.h, [18](#)
- logger.h
 - initialize_logger, [10](#)
- protocol.h
 - add_to_packet, [13](#)
 - create_buffer, [13](#)
 - create_packet, [14](#)
 - destroy_packet, [14](#)
 - fetch_buffer, [14](#)
 - fetch_codop, [15](#)
 - fetch_packet, [15](#)
 - send_packet, [15](#)
 - serialize_packet, [16](#)
- send_packet
 - protocol.h, [15](#)
- serialize_packet
 - protocol.h, [16](#)
- sockets.h
 - close_conection, [17](#)
 - create_conection, [17](#)
 - initialize_server, [18](#)
 - wait_client, [18](#)
- t_buffer, [5](#)
- t_packet, [5](#)
- t_process_conection_args, [6](#)
- utils/include/comunication.h, [7](#)
- utils/include/configs.h, [7](#)
- utils/include/logger.h, [9](#)
- utils/include/opcode.h, [11](#)
- utils/include/protocol.h, [12](#)
- utils/include/sockets.h, [16](#)
- utils/include/utils.h, [19](#)
- wait_client
 - sockets.h, [18](#)