COmmunication MOdule de SIMBA Manual de referencia

Generado por Doxygen 1.3.7

Tue Mar 29 12:57:08 2005

Índice general

Capítulo 1

COmmunication MOdule de SIMBA Indice de clases

1.1. COmmunication MOdule de SIMBA Lista de componentes

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

Capítulo 2

COmmunication MOdule de SIMBA Indice de archivos

2.1. COmmunication MOdule de SIMBA Lista de archivos

Lista de todos los archivos con descripciones breves:

Capítulo 3

COmmunication MOdule de SIMBA Documentación de clases

3.1. Referencia de la Estructura _chunk

Particion de un mensaje para ser enviado por red.

`#include <global.h>`

Atributos públicos

- int index
- int total
- int sum
- unsigned char data [CHUNK_SIZE]

3.1.1. Descripción detallada

Particion de un mensaje para ser enviado por red.

Definición en la línea 26 del archivo global.h.

3.1.2. Documentación de los datos miembro

3.1.2.1. unsigned char _chunk::data[CHUNK_SIZE]

Definición en la línea 30 del archivo global.h.

3.1.2.2. int _chunk::index

Definición en la línea 27 del archivo global.h.

3.1.2.3. int _chunk::sum

Definición en la línea 29 del archivo global.h.

3.1.2.4. int _chunk::total

Definición en la línea 28 del archivo global.h.

La documentación para esta estructura fué generada a partir del siguiente archivo:

- global.h

3.2. Referencia de la Estructura _dict_pair

Atributos públicos

- char key [TEXT_LENGTH]
- int value

3.2.1. Documentación de los datos miembro

3.2.1.1. char _dict_pair::key[TEXT_LENGTH]

Definición en la línea 34 del archivo msg_structures.c.

Referenciado por del_key().

3.2.1.2. int _dict_pair::value

Definición en la línea 35 del archivo msg_structures.c.

Referenciado por del_key(), has_key(), init_dict(), put_key(), y set_key().

La documentación para esta estructura fué generada a partir del siguiente archivo:

- msg_structures.c

3.3. Referencia de la Estructura convnode_

Nodo de los vectores de conversaciones.

`#include <msg_structures.h>`

Atributos públicos

- int index
- char name [TEXT_LENGTH]

3.3.1. Descripción detallada

Nodo de los vectores de conversaciones.

Definición en la línea 35 del archivo msg_structures.h.

3.3.2. Documentación de los datos miembro

3.3.2.1. int convnode_::index

Definición en la línea 36 del archivo msg_structures.h.

3.3.2.2. char convnode_::name[TEXT_LENGTH]

Definición en la línea 37 del archivo msg_structures.h.

La documentación para esta estructura fué generada a partir del siguiente archivo:

- msg_structures.h

3.4. Referencia de la Estructura lnode_

Nodo de una lista enlazada.

`#include <types.h>`

Atributos públicos

- char msg [TEXT_LENGTH]
- int code

3.4.1. Descripción detallada

Nodo de una lista enlazada.

Definición en la línea 20 del archivo types.h.

3.4.2. Documentación de los datos miembro

3.4.2.1. int lnode_::code

Definición en la línea 22 del archivo types.h.

Referenciado por addnode(), copynode(), getcode(), initlist(), len(), printlist(), y setcode().

3.4.2.2. char lnode_::msg[TEXT_LENGTH]

Definición en la línea 21 del archivo types.h.

La documentación para esta estructura fué generada a partir del siguiente archivo:

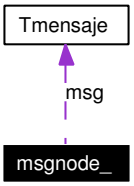
- types.h

3.5. Referencia de la Estructura msgnode_

Nodo mensaje.

`#include <msg_structures.h>`

Diagrama de colaboración para msgnode_:



Atributos públicos

- int code
- Tmensaje msg

3.5.1. Descripción detallada

Nodo mensaje.

Definición en la línea 12 del archivo msg_structures.h.

3.5.2. Documentación de los datos miembro

3.5.2.1. int msgnode_::code

Definición en la línea 13 del archivo msg_structures.h.

Referenciado por addmsgnode(), como_send_erroneous(), como_send_outbox(), delete_conv_perm(), deltopmsg(), fprint_msg_queue(), initmsglist(), print_conv(), print_msg_queue(), print_outbox(), put_in_outbox(), y updatefree().

3.5.2.2. Tmensaje msgnode_::msg

Definición en la línea 14 del archivo msg_structures.h.

Referenciado por como_send_erroneous(), como_send_outbox(), deltopmsg(), fprint_-msg_queue(), get_last_from_conv(), get_top_from_conv(), print_conv(), print_msg_-queue(), print_outbox(), y put_in_outbox().

La documentación para esta estructura fué generada a partir del siguiente archivo:

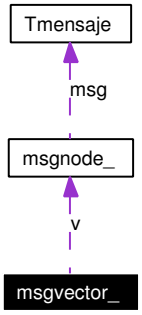
- msg_structures.h

3.6. Referencia de la Estructura msgvector_

Vector de mensajes ya parseados.

`#include <msg_structures.h>`

Diagrama de colaboración para msgvector_:



Atributos públicos

- msgnode v [MQ_SIZE]
- int firstfree
- int last
- int top

3.6.1. Descripción detallada

Vector de mensajes ya parseados.

Definición en la línea 18 del archivo msg_structures.h.

3.6.2. Documentación de los datos miembro

3.6.2.1. int msgvector_::firstfree

Definición en la línea 20 del archivo msg_structures.h.

Referenciado por addmsgnode(), delete_conv_perm(), initmsglist(), print_outbox(), put_in_outbox(), y updatefree().

3.6.2.2. int msgvector_::last

Definición en la línea 21 del archivo msg_structures.h.

Referenciado por addmsgnode(), delete_conv_perm(), deltopmsg(), get_last_from_-conv(), initmsglist(), print_outbox(), y put_in_outbox().

3.6.2.3. int msgvector_::top

Definición en la línea 22 del archivo msg_structures.h.

Referenciado por addmsgnode(), como_send_erroneous(), como_send_outbox(), delete_conv_perm(), deltopmsg(), get_top_from_conv(), initmsglist(), print_outbox(), y put_in_outbox().

3.6.2.4. msgnode msgvector_::v[MQ_SIZE]

Definición en la línea 19 del archivo msg_structures.h.

Referenciado por addmsgnode(), como_send_erroneous(), como_send_outbox(), delete_conv_perm(), deltopmsg(), fprint_msg_queue(), get_last_from_conv(), get_top_from_conv(), initmsglist(), print_conv(), print_msg_queue(), print_outbox(), put_in_outbox(), y updatefree().

La documentación para esta estructura fué generada a partir del siguiente archivo:

- msg_structures.h

3.7. Referencia de la Estructura tipo_cliente

Atributos públicos

- char direccion [TAM_DIRECCION_IP]
- int socket_cliente

3.7.1. Descripción detallada

Definicion de un nuevo tipo para guardar todos los sockets con los que esta conectado el cliente

Definición en la línea 17 del archivo transporte.c.

3.7.2. Documentación de los datos miembro

3.7.2.1. char tipo_cliente::direccion[TAM_DIRECCION_IP]

Definición en la línea 18 del archivo transporte.c.

3.7.2.2. int tipo_cliente::socket_cliente

Definición en la línea 19 del archivo transporte.c.

Referenciado por enviar_udp(), inicializar_udp(), y send_parted().

La documentación para esta estructura fué generada a partir del siguiente archivo:

- transporte.c

3.8. Referencia de la Estructura Tmensaje

`#include <mensaje.h>`

Atributos públicos

- char performative [TAM_ETIQUETA]
- char sender [TAM_ETIQUETA]
- char receiver [TAM_ETIQUETA]
- char reply_to [TAM_ETIQUETA]
- char content [TAM_CAMPO]
- char language [TAM_ETIQUETA]
- char encoding [TAM_ETIQUETA]
- char ontology [TAM_ETIQUETA]
- char protocol [TAM_ETIQUETA]
- char conversation_id [TAM_ETIQUETA]
- char reply_with [TAM_ETIQUETA]
- char in_reply_to [TAM_ETIQUETA]
- char reply_by [TAM_ETIQUETA]
- char x_subject [TAM_ETIQUETA]
- char x_priority [TAM_ETIQUETA]
- timeval tiempo_emision
- timeval tiempo_recepcion

3.8.1. Documentación de los datos miembro

3.8.1.1. char Tmensaje::content[TAM_CAMPO]

Definición en la línea 22 del archivo mensaje.h.

Referenciado por clear_msg(), copymsgnode(), extraer_campos_mensaje(), formar_mensaje(), imprimir_campos_mensaje(), inicializar_campos_mensaje(), y main().

3.8.1.2. char Tmensaje::conversation_id[TAM_ETIQUETA]

Definición en la línea 27 del archivo mensaje.h.

Referenciado por clear_msg(), como_receive(), como_receive_inbox(), como_send(), como_send_outbox(), copymsgnode(), extraer_campos_mensaje(), formar_mensaje(), imprimir_campos_mensaje(), inicializar_campos_mensaje(), main(), mi_listen(), print_outbox(), y reply_to_error().

3.8.1.3. char Tmensaje::encoding[TAM_ETIQUETA]

Definición en la línea 24 del archivo mensaje.h.

Referenciado por clear_msg(), copymsgnode(), extraer_campos_mensaje(), formar_mensaje(), imprimir_campos_mensaje(), inicializar_campos_mensaje(), y main().

3.8.1.4. char Tmensaje::in_reply_to[TAM_ETIQUETA]

Definición en la línea 29 del archivo mensaje.h.

Referenciado por clear_msg(), copymsgnode(), extraer_campos_mensaje(), formar_mensaje(), imprimir_campos_mensaje(), inicializar_campos_mensaje(), main(), y reply_to_error().

3.8.1.5. char Tmensaje::language[TAM_ETIQUETA]

Definición en la línea 23 del archivo mensaje.h.

Referenciado por clear_msg(), copymsgnode(), extraer_campos_mensaje(), formar_mensaje(), imprimir_campos_mensaje(), inicializar_campos_mensaje(), y main().

3.8.1.6. char Tmensaje::ontology[TAM_ETIQUETA]

Definición en la línea 25 del archivo mensaje.h.

Referenciado por clear_msg(), copymsgnode(), extraer_campos_mensaje(), formar_mensaje(), imprimir_campos_mensaje(), inicializar_campos_mensaje(), y main().

3.8.1.7. char Tmensaje::performative[TAM_ETIQUETA]

Definición en la línea 18 del archivo mensaje.h.

Referenciado por clear_msg(), copymsgnode(), extraer_campos_mensaje(), formar_mensaje(), fprint_msg_queue(), imprimir_campos_mensaje(), main(), print_conv(), print_msg_queue(), print_outbox(), y reply_to_error().

3.8.1.8. char Tmensaje::protocol[TAM_ETIQUETA]

Definición en la línea 26 del archivo mensaje.h.

Referenciado por clear_msg(), copymsgnode(), extraer_campos_mensaje(), formar_mensaje(), imprimir_campos_mensaje(), inicializar_campos_mensaje(), y main().

3.8.1.9. char Tmensaje::receiver[TAM_ETIQUETA]

Definición en la línea 20 del archivo mensaje.h.

Referenciado por clear_msg(), copymsgnode(), enviar(), extraer_campos_mensaje(), formar_mensaje(), fprint_msg_queue(), imprimir_campos_mensaje(), inicializar_campos_mensaje(), main(), print_conv(), print_msg_queue(), print_outbox(), y reply_to_error().

3.8.1.10. char Tmensaje::reply_by[TAM_ETIQUETA]

Definición en la línea 30 del archivo mensaje.h.

Referenciado por clear_msg(), copymsgnode(), extraer_campos_mensaje(), formar_-mensaje(), imprimir_campos_mensaje(), inicializar_campos_mensaje(), y main().

3.8.1.11. char Tmensaje::reply_to[TAM_ETIQUETA]

Definición en la línea 21 del archivo mensaje.h.

Referenciado por clear_msg(), copymsgnode(), extraer_campos_mensaje(), formar_-mensaje(), imprimir_campos_mensaje(), inicializar_campos_mensaje(), main(), y reply_to_error().

3.8.1.12. char Tmensaje::reply_with[TAM_ETIQUETA]

Definición en la línea 28 del archivo mensaje.h.

Referenciado por clear_msg(), copymsgnode(), extraer_campos_mensaje(), formar_-mensaje(), imprimir_campos_mensaje(), inicializar_campos_mensaje(), main(), y reply_to_error().

3.8.1.13. char Tmensaje::sender[TAM_ETIQUETA]

Definición en la línea 19 del archivo mensaje.h.

Referenciado por clear_msg(), copymsgnode(), extraer_campos_mensaje(), formar_-mensaje(), fprint_msg_queue(), imprimir_campos_mensaje(), inicializar_campos_-mensaje(), main(), print_conv(), print_msg_queue(), print_outbox(), recibir(), y reply_to_error().

3.8.1.14. struct timeval Tmensaje::tiempo_emision

Definición en la línea 35 del archivo mensaje.h.

Referenciado por clear_msg(), copymsgnode(), formar_mensaje(), y inicializar_-campos_mensaje().

3.8.1.15. struct timeval Tmensaje::tiempo_recepcion

Definición en la línea 36 del archivo mensaje.h.

Referenciado por clear_msg(), extraer_campos_mensaje(), y inicializar_campos_-mensaje().

3.8.1.16. char Tmensaje::x_priority[TAM_ETIQUETA]

Definición en la línea 32 del archivo mensaje.h.

Referenciado por clear_msg(), copymsgnode(), extraer_campos_mensaje(), formar_-mensaje(), imprimir_campos_mensaje(), inicializar_campos_mensaje(), main(), y reply_to_error().

3.8.1.17. char Tmensaje::x_subject[TAM_ETIQUETA]

Definición en la línea 31 del archivo mensaje.h.

Referenciado por clear_msg(), copymsgnode(), extraer_campos_mensaje(), formar_-mensaje(), imprimir_campos_mensaje(), inicializar_campos_mensaje(), y main().

La documentación para esta estructura fué generada a partir del siguiente archivo:

- mensaje.h

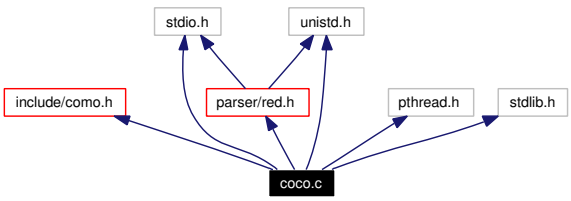
Capítulo 4

COmmunication MOdule de SIMBA Documentación de archivos

4.1. Referencia del Archivo coco.c

```
#include "include/como.h"
#include "parser/red.h"
#include <unistd.h>
#include <stdio.h>
#include <pthread.h>
#include <stdlib.h>
```

Dependencia gráfica adjunta para coco.c:



Definiciones

- #define TAMLINEA 255
- #define TAMMSG 10240

Funciones

- int read_msg_from_file (char *arg1, char *mensaje)
- void processcommand (char *c)
- void * mi_listen (void *args)
- int main ()

4.1.1. Documentación de las definiciones

4.1.1.1. #define TAMLINEA 255

Definición en la línea 10 del archivo coco.c.

Referenciado por main(), y read_msg_from_file().

4.1.1.2. #define TAMMSG 10240

Definición en la línea 11 del archivo coco.c.

Referenciado por main(), y processcommand().

4.1.2. Documentación de las funciones

4.1.2.1. int main ()

Definición en la línea 220 del archivo coco.c.

Hace referencia a como_init(), direccion_ip, processcommand(), TAM_ETIQUETA, y TAMLINEA.

4.1.2.2. void* mi_listen (void * *args*)

Definición en la línea 205 del archivo coco.c.

Hace referencia a Tmensaje::conversation_id, pm, put_msg(), y recibir().

4.1.2.3. void processcommand (char * *c*)

Definición en la línea 30 del archivo coco.c.

Hace referencia a como_accept_msg(), como_close_conversation(), como_create_conversation(), como_delete_conversation(), como_receive(), como_receive_inbox(), como_send(), como_send_erroneous(), como_send_outbox(), print_conv(), print_msg_queue(), print_outbox(), read_msg_from_file(), y TAMMSG.

Referenciado por main().

4.1.2.4. int read_msg_from_file (char * *arg1*, char * *mensaje*)

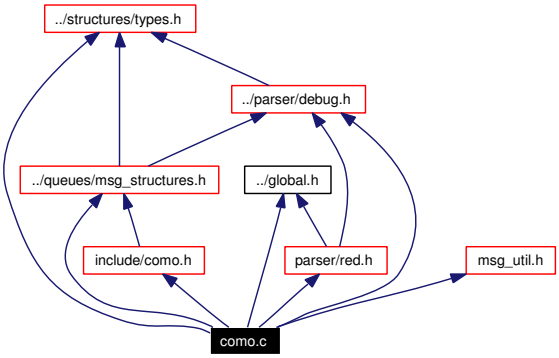
Definición en la línea 14 del archivo coco.c.

Hace referencia a fd, linea, mensaje, y TAMLINEA.

4.2. Referencia del Archivo como.c

```
#include "include/como.h"
#include "structures/types.h"
#include "parser/debug.h"
#include "parser/global.h"
#include "parser/red.h"
#include "queues/msg_structures.h"
#include "msg_util.h"
```

Dependencia gráfica adjunta para como.c:



Funciones

- int get_conv_index (char *c_id)
- int get_conv_index_no_create (char *c_id)
- void deltopmsg (int conv)
- int is_pending (int c)
- int is_active (int c)
- int is_terminated (int c)
- int delete_conv_perm (char *conv_id)
- int como_init ()
- int como_send (char *msg, char *conv_id)
- Tmensaje * como_receive (char *conv_id)
- Tmensaje * como_receive_service (char *service)
- int como_create_conversation (char *conv_id)
- int como_close_conversation (char *conv_id)
- void como_accept_msg (char *conv_id)
- int como_delete_conversation (char *conv_id)
- int como_send_erroneous (int n)
- int como_send_outbox (int n)
- int como_receive_inbox (int n)

4.2.1. Documentación de las funciones

4.2.1.1. void como_accept_msg (char * *conv_id*)

Acepta y borra efectivamente el primer mensaje de la lista de mensajes de la conversacion

Parámetros:
La conversacion de la cual se acepta el mensaje

Definición en la línea 99 del archivo como.c.

Hace referencia a deltopmsg(), y get_conv_index_no_create().

Referenciado por processcommand().

4.2.1.2. int como_close_conversation (char * *conv_id*)

Cierra una conversacion

Parámetros:
conv_id La conversacion a cerrar

Devuelve:
Codigo de error si lo hay

Definición en la línea 88 del archivo como.c.

Hace referencia a get_conv_index_no_create(), y set_terminated_conversation().

Referenciado por processcommand().

4.2.1.3. int como_create_conversation (char * *conv_id*)

Crea una nueva conversacion

Parámetros:
conv_id Identificador deseado para la conversacion

Devuelve:
Codigo de error si lo hay

Definición en la línea 84 del archivo como.c.

Hace referencia a get_conv_index().

Referenciado por processcommand().

4.2.1.4. int como_delete_conversation (char * *conv_id*)

Borra permanentemente una conversacion de la lista de Conversaciones Terminadas

Parámetros:
conv_id La conversacion

Devuelve:
Codigo de error si lo hay

Definición en la línea 109 del archivo como.c.

Hace referencia a delete_conv_perm().

Referenciado por processcommand().

4.2.1.5. int como_init ()

Inicializa el CoMo

Devuelve:
Error si lo hubiera, cero en caso contrario

Definición en la línea 19 del archivo como.c.

Hace referencia a direccion_ip, inicializar_red(), initmsglist(), obtener_mi_direccion_ip(), y TAM_DIRECCION_IP.

Referenciado por main().

4.2.1.6. Tmensaje* como_receive (char * *conv_id*)

Recibir un mensaje de una conversacion

Notese que el mensaje NO se elimina de la lista de mensajes, tan solo se devuelve una referencia al mismo

Parámetros:
conv_id Identificador de la conversacion

Devuelve:
Puntero al mensaje

Definición en la línea 56 del archivo como.c.

Hace referencia a Tmensaje::conversation_id, get_conv_index(), get_top_from_conv(), is_terminated(), y set_active_conversation().

Referenciado por como_receive_service(), y processcommand().

4.2.1.7. int como_receive_inbox (int *n*)

Recibe un numero determinado de mensajes

Parámetros:
n Numero maximo de mensajes a recibir

Devuelve:
Numero de mensajes realmente recibidos

Definición en la línea 184 del archivo como.c.

Hace referencia a Tmensaje::conversation_id, is_active(), is_pending(), put_msg(), recibir(), y set_pending_conversation().

Referenciado por processcommand().

4.2.1.8. Tmensaje* como_receive_service (char * *service*)

Recibe un mensaje relativo a un servicio que oferta el In-Agent

Parámetros:
service El servicio ofertado

Devuelve:
Puntero al mensaje

Definición en la línea 79 del archivo como.c.

Hace referencia a como_receive().

4.2.1.9. int como_send (char * *msg*, char * *conv_id*)

Enviar un mensaje FIPA

Parámetros:
msg El mensaje FIPA

Definición en la línea 35 del archivo como.c.

Hace referencia a Tmensaje::conversation_id, ERROR_NONE, extraer_campos_mensaje(), y put_in_outbox().

Referenciado por main(), y processcommand().

4.2.1.10. int como_send_erroneous (int *n*)

Contesta mensajes erroneos

Lee hasta n mensajes de la cola de mensajes erroneos, contruye sus respectivas contestaciones (NOT_UNDERSTOOD) y las envia sin pasar por la bandeja de salida. TODO: Se borran los mensajes de la cola de erroneos?

Parámetros:
n Numero maximo de mensajes erroneos a procesar

Devuelve:
Numero de mensajes erroneo atendidos

Definición en la línea 115 del archivo como.c.

Hace referencia a msgnode_::code, deltopmsg(), msgnode_::msg, msg_queue, msgnode, reply_to_error(), msgvector_::top, y msgvector_::v.

Referenciado por processcommand().

4.2.1.11. int como_send_outbox (int *n*)

Envia mensajes de la bandeja de salida

Envia hasta n mensajes de la bandeja de salida

Parámetros:

n Numero maximo de mensajes de la bandeja a procesar

Definición en la línea 144 del archivo como.c.

Hace referencia a msgnode_::code, Tmensaje::conversation_id, deltopmsg(), enviar(), get_conv_index(), msgnode_::msg, outbox, set_active_conversation(), msgvector_::top, y msgvector_::v.

Referenciado por processcommand().

4.2.1.12. int delete_conv_perm (char * *conv_id*)

Borra PERMANENTEMENTE una conversacion

Parámetros:

c El indice de la conversacion

Devuelve:

0 si todo ha ido bien, -1 si ha habido algun error

Definición en la línea 722 del archivo msg_structures.c.

Hace referencia a clear_msg(), msgnode_::code, del_conv_from_terminated(), del_key(), msgvector_::firstfree, get_conv_index_no_create(), msgvector_::last, MQ_SIZE, msg_queue, names_table, msgvector_::top, y msgvector_::v.

Referenciado por como_delete_conversation().

4.2.1.13. void deltopmsg (int *conv*)

Definición en la línea 147 del archivo msg_structures.c.

Hace referencia a clear_msg(), msgnode_::code, msgvector_::last, msgnode_::msg, msg_queue, outbox, msgvector_::top, updatefree(), y msgvector_::v.

Referenciado por como_accept_msg(), como_send_erroneous(), y como_send_outbox().

4.2.1.14. int get_conv_index (char * *c_id*)

Devuelve un indice a partir de un identificador de conversacion

Devuelve un indice a partir de un identificador de conversacion. A diferencia de 'hash_conv' lo hace usando el diccionario names_table. Si no encuentra un indice asociado, LO CREA. Si no se desea este comportamiento (por ejemplo, en 'como_close_conversation') hay que usar 'get_conv_index_no_create'.

Parámetros:
c_id Identificador de conversacion

Devuelve:
El indice o -1 si hay un error

Definición en la línea 680 del archivo msg_structures.c.

Hace referencia a has_key(), names_table, y put_key().

4.2.1.15. int get_conv_index_no_create (char * *c_id*)

Definición en la línea 694 del archivo msg_structures.c.

Hace referencia a has_key(), y names_table.

4.2.1.16. int is_active (int *c*)

Comprueba si una conversacion esta en estado activo

Parámetros:
c Identificador de conversacion

Devuelve:
1 si esta activa, 0 en caso contrario

Definición en la línea 641 del archivo msg_structures.c.

Hace referencia a active_convs, y MQ_SIZE.

Referenciado por como_receive_inbox().

4.2.1.17. int is_pending (int *c*)

Comprueba si una conversacion esta en estado pendiente

Parámetros:
c Identificador de conversacion

Devuelve:
1 si esta pendiente, 0 en caso contrario

Definición en la línea 623 del archivo msg_structures.c.

Hace referencia a MQ_SIZE, y pending_convs.

Referenciado por como_receive_inbox().

4.2.1.18. int is_terminated (int *c*)

Comprueba si una conversacion esta en estado terminado

Parámetros:

c Identificador de conversacion

Devuelve:

1 si esta terminada, o en caso contrario

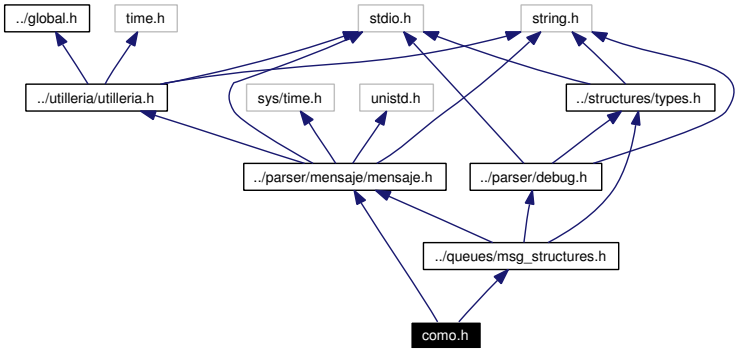
Definición en la línea 658 del archivo msg_structures.c.

Hace referencia a MQ_SIZE, y terminated_convs.

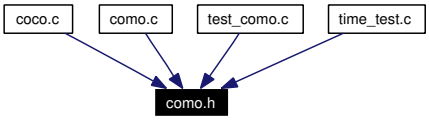
4.3. Referencia del Archivo como.h

```
#include "../parser/mensaje/mensaje.h"
#include "../queues/msg_structures.h"
```

Dependencia gráfica adjunta para como.h:



Este gráfico muestra que archivos directa o indirectamente incluyen a este archivo:



Funciones

- int como_init ()
- int como_send (char ∗msg, char ∗conv_id)
- Tmensaje ∗ como_receive (char ∗conv_id)
- Tmensaje ∗ como_receive_service (char ∗service)
- int como_create_conversation (char ∗conv_id)
- int como_close_conversation (char ∗conv_id)
- void como_accept_msg (char ∗conv_id)
- int como_delete_conversation (char ∗conv_id)
- int como_send_erroneous (int n)
- int como_send_outbox (int n)
- int como_receive_inbox (int n)

4.3.1. Documentación de las funciones

4.3.1.1. void como_accept_msg (char ∗ *conv_id*)

Acepta y borra efectivamente el primer mensaje de la lista de mensajes de la conversacion

Parámetros:
La conversacion de la cual se acepta el mensaje

Definición en la línea 99 del archivo como.c.

Hace referencia a deltopmsg(), y get_conv_index_no_create().

Referenciado por processcommand().

4.3.1.2. int como_close_conversation (char * *conv_id*)

Cierra una conversacion

Parámetros:
conv_id La conversacion a cerrar

Devuelve:
Codigo de error si lo hay

Definición en la línea 88 del archivo como.c.

Hace referencia a get_conv_index_no_create(), y set_terminated_conversation().

Referenciado por processcommand().

4.3.1.3. int como_create_conversation (char * *conv_id*)

Crea una nueva conversacion

Parámetros:
conv_id Identificador deseado para la conversacion

Devuelve:
Codigo de error si lo hay

Definición en la línea 84 del archivo como.c.

Hace referencia a get_conv_index().

Referenciado por processcommand().

4.3.1.4. int como_delete_conversation (char * *conv_id*)

Borra permanentemente una conversacion de la lista de Conversaciones Terminadas

Parámetros:
conv_id La conversacion

Devuelve:
Codigo de error si lo hay

Definición en la línea 109 del archivo como.c.

Hace referencia a delete_conv_perm().

Referenciado por processcommand().

4.3.1.5. int como_init ()

Inicializa el CoMo

Devuelve:
Error si lo hubiera, cero en caso contrario

Definición en la línea 19 del archivo como.c.

Hace referencia a direccion_ip, inicializar_red(), initmsglist(), obtener_mi_direccion_ip(), y TAM_DIRECCION_IP.

Referenciado por main().

4.3.1.6. Tmensaje* como_receive (char * *conv_id*)

Recibir un mensaje de una conversacion

Notese que el mensaje NO se elimina de la lista de mensajes, tan solo se devuelve una referencia al mismo

Parámetros:
conv_id Identificador de la conversacion

Devuelve:
Puntero al mensaje

Definición en la línea 56 del archivo como.c.

Hace referencia a Tmensaje::conversation_id, get_conv_index(), get_top_from_conv(), is_terminated(), y set_active_conversation().

Referenciado por como_receive_service(), y processcommand().

4.3.1.7. int como_receive_inbox (int *n*)

Recibe un numero determinado de mensajes

Parámetros:
n Numero maximo de mensajes a recibir

Devuelve:
Numero de mensajes realmente recibidos

Definición en la línea 184 del archivo como.c.

Hace referencia a Tmensaje::conversation_id, is_active(), is_pending(), put_msg(), recibir(), y set_pending_conversation().

Referenciado por processcommand().

4.3.1.8. Tmensaje* como_receive_service (char * *service*)

Recibe un mensaje relativo a un servicio que oferta el In-Agent

Parámetros:
service El servicio ofertado

Devuelve:
Puntero al mensaje

Definición en la línea 79 del archivo como.c.

Hace referencia a como_receive().

4.3.1.9. int como_send (char * *msg*, char * *conv_id*)

Enviar un mensaje FIPA

Parámetros:
msg El mensaje FIPA

Definición en la línea 35 del archivo como.c.

Hace referencia a Tmensaje::conversation_id, ERROR_NONE, extraer_campos_mensaje(), y put_in_outbox().

Referenciado por main(), y processcommand().

4.3.1.10. int como_send_erroneous (int *n*)

Contesta mensajes erroneos

Lee hasta n mensajes de la cola de mensajes erroneos, contruye sus respectivas contestaciones (NOT_UNDERSTOOD) y las envia sin pasar por la bandeja de salida. TODO: Se borran los mensajes de la cola de erroneos?

Parámetros:
n Numero maximo de mensajes erroneos a procesar

Devuelve:
Numero de mensajes erroneo atendidos

Definición en la línea 115 del archivo como.c.

Hace referencia a msgnode_::code, deltopmsg(), msgnode_::msg, msg_queue, msgnode, reply_to_error(), msgvector_::top, y msgvector_::v.

Referenciado por processcommand().

4.3.1.11. int como_send_outbox (int *n*)

Envia mensajes de la bandeja de salida

Envia hasta n mensajes de la bandeja de salida

Parámetros:

n Numero maximo de mensajes de la bandeja a procesar

Definición en la línea 144 del archivo como.c.

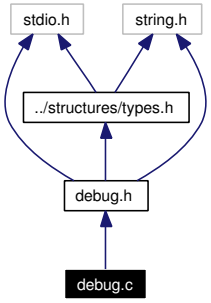
Hace referencia a msgnode_::code, Tmensaje::conversation_id, deltopmsg(), enviar(), get_conv_index(), msgnode_::msg, outbox, set_active_conversation(), msgvector_::top, y msgvector_::v.

Referenciado por processcommand().

4.4. Referencia del Archivo debug.c

```
#include "debug.h"
```

Dependencia gráfica adjunta para debug.c:



Funciones

- void blank (char msg[TEXT_LENGTH])

  Limpia una cadena de texto.

- void debug_start ()

  Prepara el programa para la depuracion.

- void debug_print (char *msg, int code)
- void debug_show_log ()

4.4.1. Documentación de las funciones

4.4.1.1. void blank (char *msg*[TEXT_LENGTH])

Limpia una cadena de texto.

Definición en la línea 11 del archivo debug.c.

Referenciado por debug_show_log().

4.4.1.2. void debug_print (char * *msg*, int *code*)

Registra un mensaje de depuracion

Inserta un mensaje de depuracion en el registro de depuracion

Parámetros:

msg El mensaje a insertar

code Codigo de depuracion asociado al mensaje

Definición en la línea 34 del archivo debug.c.

Referenciado por addmsgnode(), connectsock(), crear_socket_escucha(), enviar(), enviar_udp(), escucha_multicast(), extraer_campos_mensaje(), formar_mensaje(), imprimir_campos_mensaje(), inicializar_udp(), main(), obtener_mi_direccion_ip(), presenta_socket_enviar(), print_conv(), print_msg_queue(), print_outbox(), put_in_-outbox(), put_msg(), quita_tiempo(), receive_parted(), recibir(), send_parted(), y up-datefree().

4.4.1.3. void debug_show_log ()

Imprime el registro de depuracion

Presenta por pantalla el registro completo de depuracion con formato

Definición en la línea 50 del archivo debug.c.

Hace referencia a blank(), getcode(), getmsg(), len(), y TEXT_LENGTH.

Referenciado por main().

4.4.1.4. void debug_start ()

Prepara el programa para la depuracion.

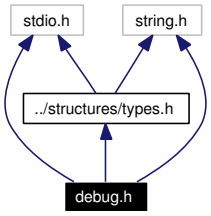
Definición en la línea 23 del archivo debug.c.

Hace referencia a addnode(), initlist(), setcode(), y setmsg().

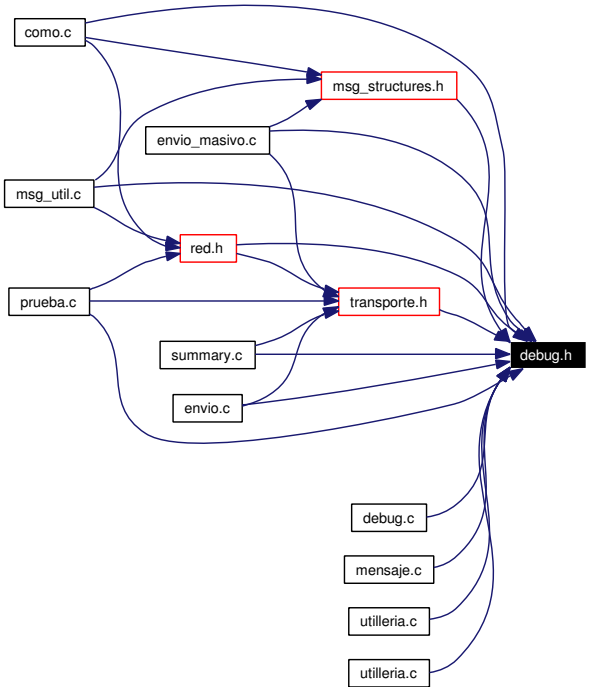
4.5. Referencia del Archivo debug.h

```
#include <stdio.h>
#include <string.h>
#include "../structures/types.h"
```

Dependencia gráfica adjunta para debug.h:



Este gráfico muestra que archivos directa o indirectamente incluyen a este archivo:



Definiciones

- #define TAM_DEBUG 10240
- #define TAM_DEBUG_LOG 10240

Funciones

- void debug_start ()

  Prepara el programa para la depuracion.

- void debug_print (char *msg, int code)
- void debug_show_log ()

4.5.1. Documentación de las definiciones

4.5.1.1. #define TAM_DEBUG 10240

Definición en la línea 9 del archivo debug.h.

Referenciado por imprimir_campos_mensaje(), y quita_tiempo().

4.5.1.2. #define TAM_DEBUG_LOG 10240

Definición en la línea 10 del archivo debug.h.

4.5.2. Documentación de las funciones

4.5.2.1. void debug_print (char * *msg*, int *code*)

Registra un mensaje de depuracion

Inserta un mensaje de depuracion en el registro de depuracion

Parámetros:

msg El mensaje a insertar

code Codigo de depuracion asociado al mensaje

Definición en la línea 34 del archivo debug.c.

Referenciado por addmsgnode(), connectsock(), crear_socket_escucha(), enviar(), enviar_udp(), escucha_multicast(), extraer_campos_mensaje(), formar_mensaje(), imprimir_campos_mensaje(), inicializar_udp(), main(), obtener_mi_direccion_ip(), presenta_socket_enviar(), print_conv(), print_msg_queue(), print_outbox(), put_in_outbox(), put_msg(), quita_tiempo(), receive_parted(), recibir(), send_parted(), y updatefree().

4.5.2.2. void debug_show_log ()

Imprime el registro de depuracion

Presenta por pantalla el registro completo de depuracion con formato

Definición en la línea 50 del archivo debug.c.

Hace referencia a blank(), getcode(), getmsg(), len(), y TEXT_LENGTH.

Referenciado por main().

4.5.2.3. void debug_start ()

Prepara el programa para la depuracion.

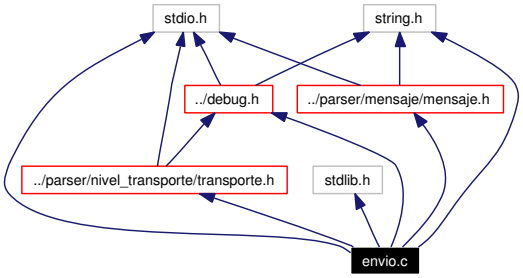
Definición en la línea 23 del archivo debug.c.

Hace referencia a addnode(), initlist(), setcode(), y setmsg().

4.6. Referencia del Archivo envio.c

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include "../parser/mensaje/mensaje.h"
#include "../parser/nivel_transporte/transporte.h"
#include "../parser/debug.h"
```

Dependencia gráfica adjunta para envio.c:



Definiciones

- #define TAMLINEA 255
- #define TAMMSG 10240

Funciones

- int main (int argc, char ∗argv[ ])

Variables

- FILE ∗ fd
- int nueva_entrada
- char linea [TAMLINEA]
- char mensaje [TAMMSG]
- Tmensaje campos

4.6.1. Documentación de las definiciones

4.6.1.1. #define TAMLINEA 255

Definición en la línea 9 del archivo envio.c.

4.6.1.2. #define TAMMSG 10240

Definición en la línea 10 del archivo envio.c.

4.6.2. Documentación de las funciones

4.6.2.1. int main (int *argc*, char * *argv*[ ])

Definición en la línea 19 del archivo envio.c.

Hace referencia a campos, Tmensaje::conversation_id, debug_print(), debug_show_-log(), enviar(), extraer_campos_mensaje(), fd, linea, mensaje, y TAMLINEA.

4.6.3. Documentación de las variables

4.6.3.1. Tmensaje campos

Definición en la línea 17 del archivo envio.c.

4.6.3.2. FILE* fd

Definición en la línea 12 del archivo envio.c.

4.6.3.3. char linea[TAMLINEA]

Definición en la línea 15 del archivo envio.c.

4.6.3.4. char mensaje[TAMMSG]

Definición en la línea 16 del archivo envio.c.

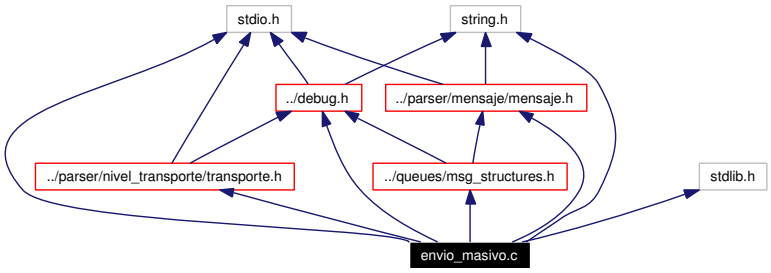
4.6.3.5. int nueva_entrada

Definición en la línea 14 del archivo envio.c.

4.7. Referencia del Archivo envio_masivo.c

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include "../parser/mensaje/mensaje.h"
#include "../parser/nivel_transporte/transporte.h"
#include "../parser/debug.h"
#include "../queues/msg_structures.h"
```

Dependencia gráfica adjunta para envio_masivo.c:



Definiciones

- #define TAMLINEA 255
- #define TAMMSG 10240

Funciones

- int main (int argc, char ∗argv[ ])

Variables

- int nueva_entrada
- char linea [TAMLINEA]
- char mensaje [TAMMSG]
- Tmensaje campos
- Tmensaje ∗ pm
- int nmesg
- char aux [TAMMSG]

4.7.1. Documentación de las definiciones

4.7.1.1. #define TAMLINEA 255

Definición en la línea 10 del archivo envio_masivo.c.

4.7.1.2. #define TAMMSG 10240

Definición en la línea 11 del archivo envio_masivo.c.

4.7.2. Documentación de las funciones

4.7.2.1. int main (int *argc*, char * *argv*[ ])

Definición en la línea 41 del archivo envio_masivo.c.

Hace referencia a aux, campos, Tmensaje::content, Tmensaje::conversation_id, debug_print(), Tmensaje::encoding, enviar(), Tmensaje::in_reply_to, Tmensaje::language, nmesg, Tmensaje::ontology, Tmensaje::performative, pm, Tmensaje::protocol, Tmensaje::receiver, Tmensaje::reply_by, Tmensaje::reply_to, Tmensaje::reply_with, Tmensaje::sender, Tmensaje::x_priority, y Tmensaje::x_subject.

4.7.3. Documentación de las variables

4.7.3.1. char aux[TAMMSG]

Definición en la línea 21 del archivo envio_masivo.c.

Referenciado por addmsgnode(), connectsock(), enviar_udp(), extraer_campos_mensaje(), fprint_msg_queue(), imprimir_campos_mensaje(), main(), presenta_socket_enviar(), print_conv(), print_msg_queue(), print_outbox(), put_in_outbox(), put_msg(), quita_tiempo(), send_parted(), y updatefree().

4.7.3.2. Tmensaje campos

Definición en la línea 18 del archivo envio_masivo.c.

4.7.3.3. char linea[TAMLINEA]

Definición en la línea 16 del archivo envio_masivo.c.

4.7.3.4. char mensaje[TAMMSG]

Definición en la línea 17 del archivo envio_masivo.c.

4.7.3.5. int nmesg

Definición en la línea 20 del archivo envio_masivo.c.

Referenciado por main().

4.7.3.6. int nueva_entrada

Definición en la línea 15 del archivo envio_masivo.c.

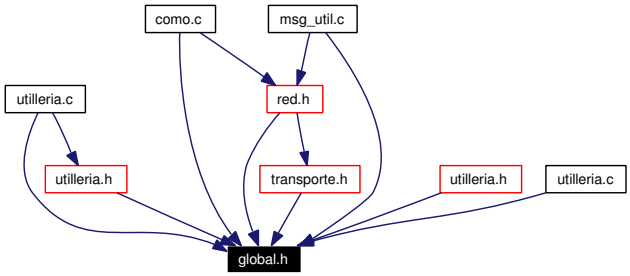
4.7.3.7. Tmensaje* pm

Definición en la línea 19 del archivo envio_masivo.c.

Referenciado por get_last_from_conv(), get_top_from_conv(), main(), y mi_listen().

4.8. Referencia del Archivo global.h

Este gráfico muestra que archivos directa o indirectamente incluyen a este archivo:



Clases

- struct _chunk

  Particion de un mensaje para ser enviado por red.

Definiciones

- #define TAM_MSG 10240
- #define MAX_ROBOTS 2
- #define TAM_DIRECCION_IP 50
- #define FALSE 0
- #define TRUE 1
- #define CHUNK_SIZE 20

  Tamanyo de particion de mensaje.

- #define MSG_MAX_SIZE 1024

Tipos definidos

- typedef _chunk chunk

  Particion de un mensaje para ser enviado por red.

4.8.1. Documentación de las definiciones

4.8.1.1. #define CHUNK_SIZE 20

Tamanyo de particion de mensaje.

Definición en la línea 18 del archivo global.h.

Referenciado por receive_parted(), send_parted(), y summarize_chunk().

4.8.1.2. #define FALSE 0

Definición en la línea 11 del archivo global.h.

Referenciado por enviar_udp(), quitar_caracteres_inutiles(), y send_parted().

4.8.1.3. #define MAX_ROBOTS 2

Definición en la línea 6 del archivo global.h.

Referenciado por enviar_udp(), inicializar_udp(), presenta_socket_enviar(), receive_parted(), y send_parted().

4.8.1.4. #define MSG_MAX_SIZE 1024

Definición en la línea 23 del archivo global.h.

Referenciado por clean_rcv_buffer(), y inicializar_udp().

4.8.1.5. #define TAM_DIRECCION_IP 50

Definición en la línea 7 del archivo global.h.

Referenciado por como_init(), main(), obtener_dir_ip_servidor(), y recibir().

4.8.1.6. #define TAM_MSG 10240

Definición en la línea 5 del archivo global.h.

Referenciado por enviar(), main(), obtener_mi_direccion_ip(), recibir(), y recibir_udp().

4.8.1.7. #define TRUE 1

Definición en la línea 14 del archivo global.h.

Referenciado por enviar_udp(), quitar_caracteres_inutiles(), y send_parted().

4.8.2. Documentación de los tipos definidos

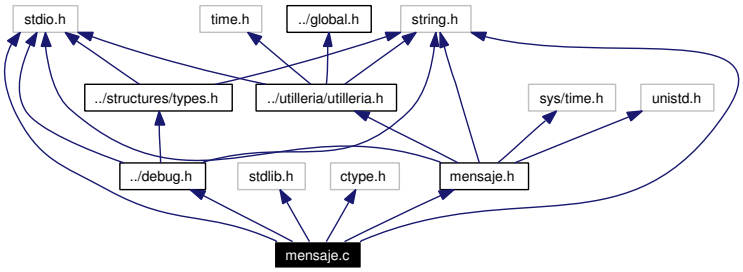
4.8.2.1. typedef struct _chunk chunk

Particion de un mensaje para ser enviado por red.

4.9. Referencia del Archivo mensaje.c

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <ctype.h>
#include "mensaje.h"
#include "../debug.h"
```

Dependencia gráfica adjunta para mensaje.c:



Definiciones

- #define TRUE 1
- #define FALSE 0

Funciones

- void inicializar_campos_mensaje (Tmensaje *campos_mensaje)
- int quita_tiempo (char *msg, long *Tv_sec, long *Tv_usec)
- int extraer_campos_mensaje (char msg[ ], Tmensaje *campos_mensaje)
- void imprimir_campos_mensaje (Tmensaje campos_mensaje)
- int formar_mensaje (Tmensaje campos_mensaje, char mensaje[TAM_CAMPO])
- void quitar_caracteres_inutiles (char msg[ ])
- void clear_msg_field (char *s)
- void clear_msg (Tmensaje m)

4.9.1. Documentación de las definiciones

4.9.1.1. #define FALSE 0

Definición en la línea 10 del archivo mensaje.c.

4.9.1.2. #define TRUE 1

Definición en la línea 9 del archivo mensaje.c.

4.9.2. Documentación de las funciones

4.9.2.1. void clear_msg (Tmensaje *m*)

Borra ("limpia") un mensaje

Parámetros:

m El mensaje

Definición en la línea 567 del archivo mensaje.c.

Hace referencia a clear_msg_field(), Tmensaje::content, Tmensaje::conversation_id, Tmensaje::encoding, Tmensaje::in_reply_to, Tmensaje::language, Tmensaje::ontology, Tmensaje::performative, Tmensaje::protocol, Tmensaje::receiver, Tmensaje::reply_by, Tmensaje::reply_to, Tmensaje::reply_with, Tmensaje::sender, Tmensaje::tiempo_emision, Tmensaje::tiempo_recepcion, Tmensaje::x_priority, y Tmensaje::x_subject.

Referenciado por delete_conv_perm(), y deltopmsg().

4.9.2.2. void clear_msg_field (char * *s*)

Borra un campo de mensaje

Parámetros:

s El campo (cadena de caracteres)

Definición en la línea 558 del archivo mensaje.c.

Hace referencia a TAM_ETIQUETA.

Referenciado por clear_msg().

4.9.2.3. int extraer_campos_mensaje (char *msg*[ ], Tmensaje * *campos_mensaje*)

Extrae campos de mensaje desde una cadena

Funcion que dado un mensaje contenido en una cadena (msg) extrae todos sus campos y los devuelve en la estructura campos_mensaje

Parámetros:

msg El mensaje a analizar

campos_mensaje La estructura a rellenar

Devuelve:

0 si todo funciono correctamente, -1 en otro caso

Definición en la línea 124 del archivo mensaje.c.

Hace referencia a aux, Tmensaje::content, Tmensaje::conversation_id, debug_print(), Tmensaje::encoding, Tmensaje::in_reply_to, inicializar_campos_mensaje(), Tmensaje::language, Tmensaje::ontology, Tmensaje::performative, Tmensaje::protocol, quita_tiempo(), quitar_caracteres_inutiles(), Tmensaje::receiver, Tmensaje::reply_by, Tmensaje::reply_to, Tmensaje::reply_with, Tmensaje::sender, TAM_CAMPO, TAM_ETIQUETA, TEXT_LENGTH, Tmensaje::tiempo_recepcion, Tmensaje::x_priority, y Tmensaje::x_subject.

Referenciado por como_send(), main(), y recibir().

4.9.2.4. int formar_mensaje (Tmensaje *campos_mensaje*, char *mensaje*[TAM_CAMPO])

Forma un mensaje a partir de una estructura

Funcion que a partir de los campos del mensaje forma una cadena. Ademas quita la repeticion de todos los espacios en blanco, caracteres de nueva linea y tabuladores.

Parámetros:

campos_mensaje La estructura con el mensaje

mensaje El mensaje a analizar

Devuelve:

0 si todo funciono correctamente, -1 en otro caso

Definición en la línea 417 del archivo mensaje.c.

Hace referencia a Tmensaje::content, Tmensaje::conversation_id, debug_print(), Tmensaje::encoding, Tmensaje::in_reply_to, Tmensaje::language, mensaje, Tmensaje::ontology, Tmensaje::performative, Tmensaje::protocol, quitar_caracteres_inutiles(), Tmensaje::receiver, Tmensaje::reply_by, Tmensaje::reply_to, Tmensaje::reply_with, Tmensaje::sender, Tmensaje::tiempo_emision, Tmensaje::x_priority, y Tmensaje::x_subject.

Referenciado por enviar(), y main().

4.9.2.5. void imprimir_campos_mensaje (Tmensaje *campos_mensaje*)

Imprime los campos de un mensaje

Funcion q presenta los distintos campos q hay rellenados de un mensaje

Parámetros:

campos_mensaje La estructura con el mensaje

Definición en la línea 332 del archivo mensaje.c.

Hace referencia a aux, Tmensaje::content, Tmensaje::conversation_id, debug_print(), Tmensaje::encoding, Tmensaje::in_reply_to, Tmensaje::language, Tmensaje::ontology, Tmensaje::performative, Tmensaje::protocol, Tmensaje::receiver,

Tmensaje::reply_by, Tmensaje::reply_to, Tmensaje::reply_with, Tmensaje::sender, TAM_DEBUG, Tmensaje::x_priority, y Tmensaje::x_subject.

Referenciado por main(), y recibir().

4.9.2.6. void inicializar_campos_mensaje (Tmensaje * *campos_mensaje*)

Inicializa una estructura de mensaje

Funcion que dada una estructura de tipo mensaje la inicializa

Parámetros:

campos_mensaje La estructura de mensaje

Definición en la línea 13 del archivo mensaje.c.

Hace referencia a Tmensaje::content, Tmensaje::conversation_id, Tmensaje::encoding, Tmensaje::in_reply_to, Tmensaje::language, Tmensaje::ontology, Tmensaje::protocol, Tmensaje::receiver, Tmensaje::reply_by, Tmensaje::reply_to, Tmensaje::reply_with, Tmensaje::sender, Tmensaje::tiempo_emision, Tmensaje::tiempo_recepcion, Tmensaje::x_priority, y Tmensaje::x_subject.

Referenciado por extraer_campos_mensaje().

4.9.2.7. int quita_tiempo (char * *msg*, long * *Tv_sec*, long * *Tv_usec*)

Quita la marca de tiempo de un mensaje

Funcion que dado un mensaje le quita el tiempo de la cabecera y lo devuelve dentro de la estructura de tiempos

Parámetros:

msg Cadena con el mensaje

Tv_sec Tiempo de vida en segundos (long)

Tv_sec Tiempo de vida en microsegundos (long)

Definición en la línea 47 del archivo mensaje.c.

Hace referencia a aux, debug_print(), ERROR_NONE, TAM_CAMPO, TAM_DEBUG, y TAM_ETIQUETA.

Referenciado por extraer_campos_mensaje().

4.9.2.8. void quitar_caracteres_inutiles (char *msg*[ ])

Retira caracteres inutiles

Funcion que quita todos los caracteres inutiles de una cadena de mensaje, como son los retornos de carro, los tabuladores y los espacios sucesivos

Parámetros:

msg Cadena con el mensaje

Definición en la línea 519 del archivo mensaje.c.

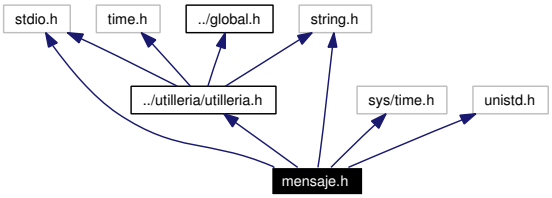
Hace referencia a FALSE, TAM_CAMPO, y TRUE.

Referenciado por extraer_campos_mensaje(), formar_mensaje(), y main().

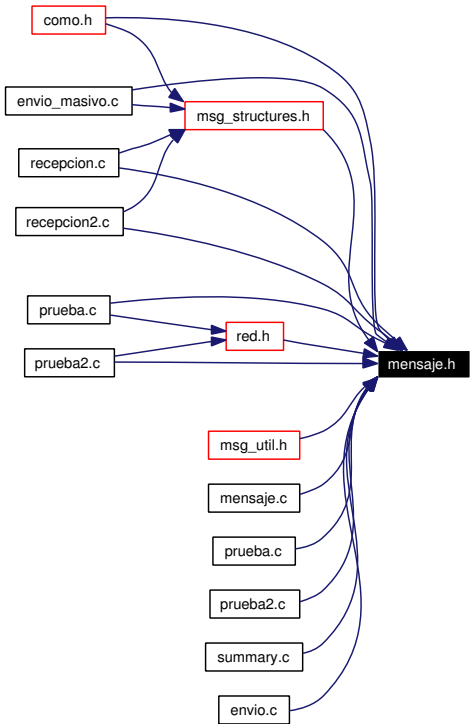
4.10. Referencia del Archivo mensaje.h

```
#include <stdio.h>
#include <string.h>
#include <sys/time.h>
#include <unistd.h>
#include "../utilleria/utilleria.h"
```

Dependencia gráfica adjunta para mensaje.h:



Este gráfico muestra que archivos directa o indirectamente incluyen a este archivo:



Clases

- struct Tmensaje

Definiciones

- #define TAM_ETIQUETA 10240
- #define TAM_CAMPO 10240

Funciones

- int extraer_campos_mensaje (char msg[ ], Tmensaje *campos_mensaje)
- int formar_mensaje (Tmensaje campos_mensaje, char mensaje[TAM_CAMPO])
- void imprimir_campos_mensaje (Tmensaje campos_mensaje)
- void quitar_caracteres_inutiles (char msg[ ])
- void inicializar_campos_mensaje (Tmensaje *campos_mensaje)
- void clear_msg (Tmensaje m)

4.10.1. Documentación de las definiciones

4.10.1.1. #define TAM_CAMPO 10240

Definición en la línea 14 del archivo mensaje.h.

Referenciado por extraer_campos_mensaje(), quita_tiempo(), y quitar_caracteres_inutiles().

4.10.1.2. #define TAM_ETIQUETA 10240

Definición en la línea 13 del archivo mensaje.h.

Referenciado por clear_msg_field(), extraer_campos_mensaje(), main(), y quita_tiempo().

4.10.2. Documentación de las funciones

4.10.2.1. void clear_msg (Tmensaje *m*)

Borra ("limpia") un mensaje

Parámetros:

m El mensaje

Definición en la línea 567 del archivo mensaje.c.

Hace referencia a clear_msg_field(), Tmensaje::content, Tmensaje::conversation_id, Tmensaje::encoding, Tmensaje::in_reply_to, Tmensaje::language, Tmensaje::ontology, Tmensaje::performative, Tmensaje::protocol, Tmensaje::receiver, Tmensaje::reply_by, Tmensaje::reply_to, Tmensaje::reply_with, Tmensaje::sender, Tmensaje::tiempo_emision, Tmensaje::tiempo_recepcion, Tmensaje::x_priority, y Tmensaje::x_subject.

Referenciado por delete_conv_perm(), y deltopmsg().

4.10.2.2. int extraer_campos_mensaje (char *msg*[ ], Tmensaje * *campos_mensaje*)

Extrae campos de mensaje desde una cadena

Funcion que dado un mensaje contenido en una cadena (msg) extrae todos sus campos y los devuelve en la estructura campos_mensaje

Parámetros:

msg El mensaje a analizar

campos_mensaje La estructura a rellenar

Devuelve:

0 si todo funciono correctamente, -1 en otro caso

Definición en la línea 124 del archivo mensaje.c.

Hace referencia a aux, Tmensaje::content, Tmensaje::conversation_id, debug_print(), Tmensaje::encoding, Tmensaje::in_reply_to, inicializar_campos_mensaje(), Tmensaje::language, Tmensaje::ontology, Tmensaje::performative, Tmensaje::protocol, quita_tiempo(), quitar_caracteres_inutiles(), Tmensaje::receiver, Tmensaje::reply_by, Tmensaje::reply_to, Tmensaje::reply_with, Tmensaje::sender, TAM_CAMPO, TAM_ETIQUETA, TEXT_LENGTH, Tmensaje::tiempo_recepcion, Tmensaje::x_priority, y Tmensaje::x_subject.

Referenciado por como_send(), main(), y recibir().

4.10.2.3. int formar_mensaje (Tmensaje *campos_mensaje*, char *mensaje*[TAM_CAMPO])

Forma un mensaje a partir de una estructura

Funcion que a partir de los campos del mensaje forma una cadena. Ademas quita la repeticion de todos los espacios en blanco, caracteres de nueva linea y tabuladores.

Parámetros:

campos_mensaje La estructura con el mensaje

mensaje El mensaje a analizar

Devuelve:

0 si todo funciono correctamente, -1 en otro caso

Definición en la línea 417 del archivo mensaje.c.

Hace referencia a Tmensaje::content, Tmensaje::conversation_id, debug_print(), Tmensaje::encoding, Tmensaje::in_reply_to, Tmensaje::language, mensaje, Tmensaje::ontology, Tmensaje::performative, Tmensaje::protocol, quitar_caracteres_inutiles(), Tmensaje::receiver, Tmensaje::reply_by, Tmensaje::reply_to, Tmensaje::reply_with, Tmensaje::sender, Tmensaje::tiempo_emision, Tmensaje::x_priority, y Tmensaje::x_subject.

Referenciado por enviar(), y main().

4.10.2.4. void imprimir_campos_mensaje (Tmensaje *campos_mensaje*)

Imprime los campos de un mensaje

Funcion q presenta los distintos campos q hay rellenados de un mensaje

Parámetros:

campos_mensaje La estructura con el mensaje

Definición en la línea 332 del archivo mensaje.c.

Hace referencia a aux, Tmensaje::content, Tmensaje::conversation_id, debug_print(), Tmensaje::encoding, Tmensaje::in_reply_to, Tmensaje::language, Tmensaje::ontology, Tmensaje::performative, Tmensaje::protocol, Tmensaje::receiver, Tmensaje::reply_by, Tmensaje::reply_to, Tmensaje::reply_with, Tmensaje::sender, TAM_DEBUG, Tmensaje::x_priority, y Tmensaje::x_subject.

Referenciado por main(), y recibir().

4.10.2.5. void inicializar_campos_mensaje (Tmensaje * *campos_mensaje*)

Inicializa una estructura de mensaje

Funcion que dada una estructura de tipo mensaje la inicializa

Parámetros:

campos_mensaje La estructura de mensaje

Definición en la línea 13 del archivo mensaje.c.

Hace referencia a Tmensaje::content, Tmensaje::conversation_id, Tmensaje::encoding, Tmensaje::in_reply_to, Tmensaje::language, Tmensaje::ontology, Tmensaje::protocol, Tmensaje::receiver, Tmensaje::reply_by, Tmensaje::reply_to, Tmensaje::reply_with, Tmensaje::sender, Tmensaje::tiempo_emision, Tmensaje::tiempo_recepcion, Tmensaje::x_priority, y Tmensaje::x_subject.

Referenciado por extraer_campos_mensaje().

4.10.2.6. void quitar_caracteres_inutiles (char *msg*[ ])

Retira caracteres inutiles

Funcion que quita todos los caracteres inutiles de una cadena de mensaje, como son los retornos de carro, los tabuladores y los espacios sucesivos

Parámetros:

msg Cadena con el mensaje

Definición en la línea 519 del archivo mensaje.c.

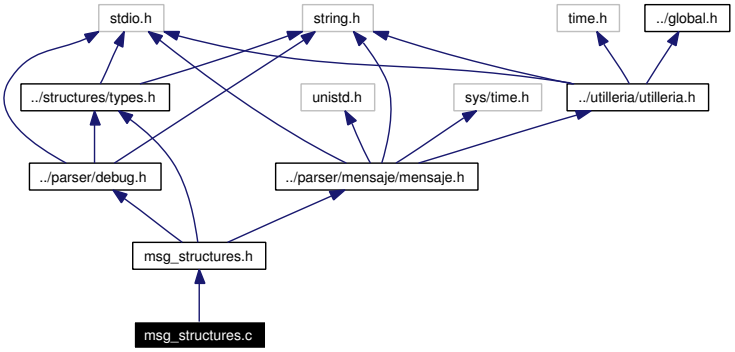
Hace referencia a FALSE, TAM_CAMPO, y TRUE.

Referenciado por extraer_campos_mensaje(), formar_mensaje(), y main().

4.11. Referencia del Archivo msg_structures.c

```
#include "msg_structures.h"
```

Dependencia gráfica adjunta para msg_structures.c:



Clases

- struct _dict_pair

Definiciones

- #define DICT_SIZE MQ_SIZE

Tipos definidos

- typedef _dict_pair dict_pair
- typedef dict_pair dict [DICT_SIZE]

Funciones

- long summarize (char *data)
- int get_conv_index (char *c_id)
- int get_conv_index_no_create (char *c_id)
- int is_terminated (int c)
- int del_conv_from_terminated (int c)
- void init_dict (dict d)
- void init_conv_list (int *cl)
- void initmsglist ()

  Inicializa la cola de mensajes y las de conversaciones.

- void updatefree (int conv)
- void copymsgnode (Tmensaje *dest, Tmensaje *orig)

- void deltopmsg (int conv)
- void addmsgnode (Tmensaje *m, int conv)
- int hash_conv (char *c_id)
- int put_msg (Tmensaje *m, char *conv_id)
- void print_outbox ()

  Imprime la bandeja de salida.

- void print_conv (char *conv_id)
- void print_msg_queue ()

  Muestra el contenido de la cola de mensajes.

- void fprint_msg_queue ()

  Imprime el contenido de la cola de mensajes en un fichero.

- int has_key (dict d, char *c_id)
- int put_key (dict d, char *c_id)
- int set_key (dict d, char *key, int value)
- int del_key (dict d, char *key)
- int remove_conversation_from_vector (int c, int *vector)
- int set_active_conversation (int c)
- int set_terminated_conversation (int c)
- int set_pending_conversation (int c)
- int is_pending (int c)
- int is_active (int c)
- Tmensaje * get_top_from_conv (int c)
- Tmensaje * get_last_from_conv (int c)
- int delete_conv_perm (char *conv_id)
- int put_in_outbox (Tmensaje *msg)

Variables

- msgvector the_msg_queue [MQ_SIZE]

  Instancia efectiva de "msg_queue".

- msgvector * msg_queue
- int active_convs [MQ_SIZE]
- int pending_convs [MQ_SIZE]
- int terminated_convs [MQ_SIZE]
- dict names_table

  Diccionario que asocia un nombre de conversacion a un indice.

4.11.1. Documentación de las definiciones

4.11.1.1. #define DICT_SIZE MQ_SIZE

Definición en la línea 4 del archivo msg_structures.c.

Referenciado por has_key(), init_dict(), y put_key().

4.11.2. Documentación de los tipos definidos

4.11.2.1. typedef dict_pair dict[DICT_SIZE]

Definición en la línea 38 del archivo msg_structures.c.

4.11.2.2. typedef struct _dict_pair dict_pair

4.11.3. Documentación de las funciones

4.11.3.1. void addmsgnode (Tmensaje * *m*, int *conv*)

Anyade un mensaje a una conversacion

Parámetros:

m El mensaje

conv La conversacion

Definición en la línea 192 del archivo msg_structures.c.

Hace referencia a aux, msgnode_::code, copymsgnode(), debug_print(), msgvector_::firstfree, msgvector_::last, msg_queue, TEXT_LENGTH, msgvector_::top, updatefree(), y msgvector_::v.

Referenciado por put_msg().

4.11.3.2. void copymsgnode (Tmensaje * *dest*, Tmensaje * *orig*)

Copia un mensaje

Parámetros:

dest Mensaje de destino

orig Mensaje de origen

Definición en la línea 122 del archivo msg_structures.c.

Hace referencia a Tmensaje::content, Tmensaje::conversation_id, Tmensaje::encoding, Tmensaje::in_reply_to, Tmensaje::language, Tmensaje::ontology, Tmensaje::performative, Tmensaje::protocol, Tmensaje::receiver, Tmensaje::reply_by,

Tmensaje::reply_to, Tmensaje::reply_with, Tmensaje::sender, Tmensaje::tiempo_-emision, Tmensaje::x_priority, y Tmensaje::x_subject.

Referenciado por addmsgnode(), y put_in_outbox().

4.11.3.3. int del_conv_from_terminated (int *c*)

Borra una conversacion de Conversaciones Terminadas

Parámetros:

c El indice de la conversacion

Devuelve:

0 si todo ha ido bien, -1 si la conversacion no estaba en el vector

Definición en la línea 755 del archivo msg_structures.c.

Hace referencia a MQ_SIZE, y terminated_convs.

Referenciado por delete_conv_perm().

4.11.3.4. int del_key (dict *d*, char * *key*)

Borra una clave y su valor de un diccionario

Parámetros:

d El diccionario

key La clave

Devuelve:

-1 en caso de error, 0 en caso contrario

Definición en la línea 512 del archivo msg_structures.c.

Hace referencia a has_key(), _dict_pair::key, TEXT_LENGTH, y _dict_pair::value.

Referenciado por delete_conv_perm().

4.11.3.5. int delete_conv_perm (char * *conv_id*)

Borra PERMANENTEMENTE una conversacion

Parámetros:

c El indice de la conversacion

Devuelve:

0 si todo ha ido bien, -1 si ha habido algun error

Definición en la línea 722 del archivo msg_structures.c.

Hace referencia a clear_msg(), msgnode_::code, del_conv_from_terminated(), del_key(), msgvector_::firstfree, get_conv_index_no_create(), msgvector_::last, MQ_SIZE, msg_queue, names_table, msgvector_::top, y msgvector_::v.

Referenciado por como_delete_conversation().

4.11.3.6. void deltopmsg (int *conv*)

Definición en la línea 147 del archivo msg_structures.c.

Hace referencia a clear_msg(), msgnode_::code, msgvector_::last, msgnode_::msg, msg_queue, outbox, msgvector_::top, updatefree(), y msgvector_::v.

Referenciado por como_accept_msg(), como_send_erroneous(), y como_send_outbox().

4.11.3.7. void fprint_msg_queue ()

Imprime el contenido de la cola de mensajes en un fichero.

Definición en la línea 399 del archivo msg_structures.c.

Hace referencia a aux, msgnode_::code, MQ_SIZE, msgnode_::msg, msg_queue, Tmensaje::performative, Tmensaje::receiver, Tmensaje::sender, TEXT_LENGTH, y msgvector_::v.

Referenciado por main().

4.11.3.8. int get_conv_index (char * *c_id*)

Devuelve un indice a partir de un identificador de conversacion

Devuelve un indice a partir de un identificador de conversacion. A diferencia de 'hash_conv' lo hace usando el diccionario names_table. Si no encuentra un indice asociado, LO CREA. Si no se desea este comportamiento (por ejemplo, en 'como_close_conversation') hay que usar 'get_conv_index_no_create'.

Parámetros:

c_id Identificador de conversacion

Devuelve:

El indice o -1 si hay un error

Referenciado por como_create_conversation(), como_receive(), como_send_outbox(), y put_msg().

4.11.3.9. int get_conv_index_no_create (char * *c_id*)

Referenciado por como_accept_msg(), como_close_conversation(), delete_conv_perm(), y print_conv().

4.11.3.10. Tmensaje* get_last_from_conv (int *c*)

Devuelve un puntero al ultimo mensaje de una conversacion

Parámetros:
c Indice de la conversacion en "msg_queue"

Devuelve:
El puntero al mensaje

Definición en la línea 709 del archivo msg_structures.c.

Hace referencia a msgvector_::last, msgnode_::msg, msg_queue, pm, y msgvector_::v.

4.11.3.11. Tmensaje* get_top_from_conv (int *c*)

Devuelve un puntero al primer mensaje efectivo de una conversacion

Parámetros:
c Indice de la conversacion en "msg_queue"

Devuelve:
El puntero al mensaje

Definición en la línea 698 del archivo msg_structures.c.

Hace referencia a msgnode_::msg, msg_queue, pm, msgvector_::top, y msgvector_::v.

Referenciado por como_receive().

4.11.3.12. int has_key (dict *d*, char * *c_id*)

Busca una clave (cadena de caracteres) en un diccionario

Parámetros:
d El diccionario

c_id La clave

Devuelve:
El valor de esa clave o -1 en caso de que la clave no exista

Definición en la línea 456 del archivo msg_structures.c.

Hace referencia a DICT_SIZE, y _dict_pair::value.

Referenciado por del_key(), get_conv_index(), get_conv_index_no_create(), y set_key().

4.11.3.13. int hash_conv (char * *c_id*)

Hash segun identificador de conversacion

Funcion de hash que devuelve un indice para la msg_queue basado en el 'conversation_id' del mensaje. Si el 'conversation_id' es '#COMO_ERROR#' devuelve siempre 0

Parámetros:
c_id 'conversation_id' del mensaje

Devuelve:
Indice para msg_queue

Definición en la línea 223 del archivo msg_structures.c.

Hace referencia a MQ_SIZE, y summarize().

4.11.3.14. void init_conv_list (int * *cl*)

Definición en la línea 45 del archivo msg_structures.c.

Hace referencia a MQ_SIZE.

Referenciado por initmsglist().

4.11.3.15. void init_dict (dict *d*)

Inicializa un diccionario

Parámetros:
d El diccionario

Definición en la línea 437 del archivo msg_structures.c.

Hace referencia a DICT_SIZE, y _dict_pair::value.

Referenciado por initmsglist().

4.11.3.16. void initmsglist ()

Inicializa la cola de mensajes y las de conversaciones.

Definición en la línea 53 del archivo msg_structures.c.

Hace referencia a active_convs, msgnode_::code, msgvector_::firstfree, init_conv_list(), init_dict(), msgvector_::last, MQ_SIZE, msg_queue, names_table, outbox, pending_convs, terminated_convs, the_msg_queue, msgvector_::top, y msgvector_::v.

Referenciado por como_init(), y main().

4.11.3.17. int is_active (int *c*)

Comprueba si una conversacion esta en estado activo

Parámetros:
c Identificador de conversacion

Devuelve:
1 si esta activa, 0 en caso contrario

Definición en la línea 641 del archivo msg_structures.c.

Hace referencia a active_convs, y MQ_SIZE.

Referenciado por como_receive_inbox().

4.11.3.18. int is_pending (int *c*)

Comprueba si una conversacion esta en estado pendiente

Parámetros:
c Identificador de conversacion

Devuelve:
1 si esta pendiente, 0 en caso contrario

Definición en la línea 623 del archivo msg_structures.c.

Hace referencia a MQ_SIZE, y pending_convs.

Referenciado por como_receive_inbox().

4.11.3.19. int is_terminated (int *c*)

Comprueba si una conversacion esta en estado terminado

Parámetros:
c Identificador de conversacion

Devuelve:
1 si esta terminada, o en caso contrario

Referenciado por como_receive(), y put_msg().

4.11.3.20. void print_conv (char * *conv_id*)

Imprime una conversacion

Parámetros:
conv_id La conversacion

Definición en la línea 303 del archivo msg_structures.c.

Hace referencia a aux, msgnode_::code, debug_print(), get_conv_index_no_create(), MQ_SIZE, msgnode_::msg, msg_queue, Tmensaje::performative, Tmensaje::receiver, Tmensaje::sender, TEXT_LENGTH, y msgvector_::v.

Referenciado por processcommand().

4.11.3.21. void print_msg_queue ()

Muestra el contenido de la cola de mensajes.

Definición en la línea 334 del archivo msg_structures.c.

Hace referencia a active_convs, aux, msgnode_::code, debug_print(), MQ_SIZE, msgnode_::msg, msg_queue, pending_convs, Tmensaje::performative, print_outbox(), Tmensaje::receiver, Tmensaje::sender, terminated_convs, TEXT_LENGTH, y msgvector_::v.

Referenciado por main(), y processcommand().

4.11.3.22. void print_outbox ()

Imprime la bandeja de salida.

Definición en la línea 274 del archivo msg_structures.c.

Hace referencia a aux, msgnode_::code, Tmensaje::conversation_id, debug_print(), msgvector_::firstfree, msgvector_::last, MQ_SIZE, msgnode_::msg, outbox, Tmensaje::performative, Tmensaje::receiver, Tmensaje::sender, TEXT_LENGTH, msgvector_::top, y msgvector_::v.

Referenciado por print_msg_queue(), y processcommand().

4.11.3.23. int put_in_outbox (Tmensaje * *msg*)

Pone un mensaje en la bandeja de salida

Parámetros:

msg El mensaje

Devuelve:

Codigo de error

Definición en la línea 768 del archivo msg_structures.c.

Hace referencia a aux, msgnode_::code, copymsgnode(), debug_print(), msgvector_::firstfree, msgvector_::last, msgnode_::msg, outbox, TEXT_LENGTH, msgvector_::top, updatefree(), y msgvector_::v.

Referenciado por como_send().

4.11.3.24. int put_key (dict *d*, char * *c_id*)

Inserta una clave en un diccionario y le pone de valor inicial su indice interno del diccionario

Parámetros:

d El diccionario

c_id La clave

Devuelve:

El indice de la clave en el diccionario o -1 si no caben mas claves

Definición en la línea 473 del archivo msg_structures.c.

Hace referencia a DICT_SIZE, y _dict_pair::value.

Referenciado por get_conv_index().

4.11.3.25. int put_msg (Tmensaje * *m*, char * *conv_id*)

Deposita un mensaje recien parseado en la cola

Parámetros:

m El mensaje

conv_id El identificador de conversacion

Devuelve:

Indice de la conversacion en 'msg_queue' o codigo de error (<0)

Definición en la línea 239 del archivo msg_structures.c.

Hace referencia a addmsgnode(), aux, debug_print(), get_conv_index(), is_terminated(), y TEXT_LENGTH.

Referenciado por como_receive_inbox(), main(), y mi_listen().

4.11.3.26. int remove_conversation_from_vector (int *c*, int * *vector*)

Quita la referencia a una conversacion en un vector de conversaciones

Parámetros:

c El indice de la conversacion en "msg_queue"

vector El vector de conversacion puede ser "pending_convs", "active_convs" o "terminated_convs"

Devuelve:

El indice que ocupaba la conversacion en el vector o -1 en caso de que no estuviera

Definición en la línea 531 del archivo msg_structures.c.

Hace referencia a MQ_SIZE.

Referenciado por set_active_conversation(), y set_terminated_conversation().

4.11.3.27. int set_active_conversation (int *c*)

Pone una conversacion en la lista de Conversaciones Activas

Parámetros:

c El indice de la conversacion en la cola de mensajes 'msg_queue'

Definición en la línea 546 del archivo msg_structures.c.

Hace referencia a active_convs, ERROR_NONE, MQ_SIZE, pending_convs, y remove_conversation_from_vector().

Referenciado por como_receive(), y como_send_outbox().

4.11.3.28. int set_key (dict *d*, char * *key*, int *value*)

Asocia un valor (entero) a una clave en un diccionario

Parámetros:

d El diccionario

key La clave

value El valor (entero)

Definición en la línea 494 del archivo msg_structures.c.

Hace referencia a has_key(), y _dict_pair::value.

4.11.3.29. int set_pending_conversation (int *c*)

Pone una conversacion en la lista de Conversaciones Pendientes

Parámetros:

c El indice de la conversacion en la cola de mensajes 'msg_queue'

Definición en la línea 598 del archivo msg_structures.c.

Hace referencia a ERROR_NONE, MQ_SIZE, y pending_convs.

Referenciado por como_receive_inbox().

4.11.3.30. int set_terminated_conversation (int *c*)

Pone una conversacion en la lista de Conversaciones Terminadas

Parámetros:

c El indice de la conversacion en la cola de mensajes 'msg_queue'

Definición en la línea 572 del archivo msg_structures.c.

Hace referencia a active_convs, ERROR_NONE, MQ_SIZE, remove_conversation_-from_vector(), y terminated_convs.

Referenciado por como_close_conversation().

4.11.3.31. long summarize (char * *data*)

Devuelve un numero resumen de una cadena de texto

Parámetros:
data La cadena a resumir

Devuelve:
Numero resumen

Definición en la línea 38 del archivo nivel_transporte/utilleria.c.

Referenciado por enviar_udp(), hash_conv(), y receive_parted().

4.11.3.32. void updatefree (int *conv*)

Actualiza el valor 'firstfree' de una conversacion

Parámetros:
conv La conversacion

Definición en la línea 94 del archivo msg_structures.c.

Hace referencia a aux, msgnode_::code, debug_print(), msgvector_::firstfree, MQ_SIZE, msg_queue, outbox, TEXT_LENGTH, y msgvector_::v.

Referenciado por addmsgnode(), deltopmsg(), y put_in_outbox().

4.11.4. Documentación de las variables

4.11.4.1. int active_convs[MQ_SIZE]

Conversaciones activas

Vector de enteros que hacen referencia a los indices de la cola de mensajes

Definición en la línea 20 del archivo msg_structures.c.

Referenciado por initmsglist(), is_active(), print_msg_queue(), set_active_conversation(), y set_terminated_conversation().

4.11.4.2. msgvector* msg_queue

Cola de conversaciones

Cola de conversaciones, compuesta de vectores de mensajes. La posicion 0 hace referencia a los mensajes erroneos

Definición en la línea 15 del archivo msg_structures.c.

Referenciado por addmsgnode(), como_send_erroneous(), delete_conv_perm(), deltopmsg(), fprint_msg_queue(), get_last_from_conv(), get_top_from_conv(), initmsglist(), print_conv(), print_msg_queue(), y updatefree().

4.11.4.3. dict names_table

Diccionario que asocia un nombre de conversacion a un indice.

Definición en la línea 41 del archivo msg_structures.c.

Referenciado por delete_conv_perm(), get_conv_index(), get_conv_index_no_-create(), y initmsglist().

4.11.4.4. int pending_convs[MQ_SIZE]

Conversaciones pendientes

Vector de enteros que hacen referencia a los indices de la cola de mensajes

Definición en la línea 25 del archivo msg_structures.c.

Referenciado por initmsglist(), is_pending(), print_msg_queue(), set_active_-conversation(), y set_pending_conversation().

4.11.4.5. int terminated_convs[MQ_SIZE]

Conversaciones acabadas

Vector de enteros que hacen referencia a los indices de la cola de mensajes

Definición en la línea 30 del archivo msg_structures.c.

Referenciado por del_conv_from_terminated(), initmsglist(), is_terminated(), print_-msg_queue(), y set_terminated_conversation().

4.11.4.6. msgvector the_msg_queue[MQ_SIZE]

Instancia efectiva de "msg_queue".

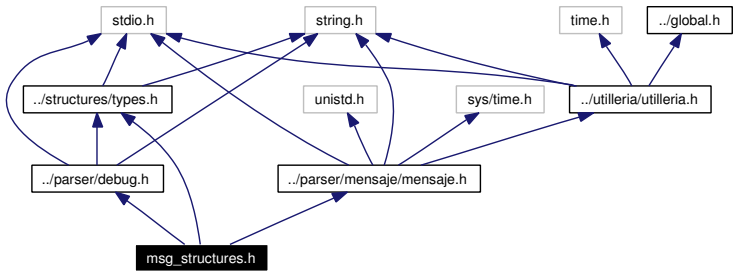
Definición en la línea 14 del archivo msg_structures.c.

Referenciado por initmsglist().

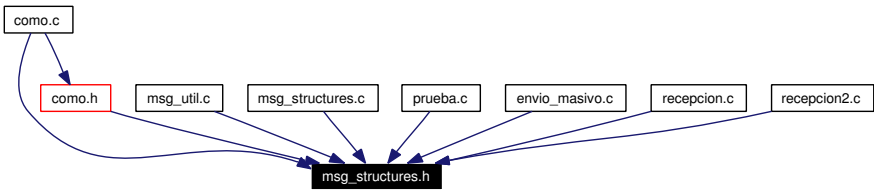
4.12. Referencia del Archivo msg_structures.h

```
#include "../parser/debug.h"
#include "../structures/types.h"
#include "../parser/mensaje/mensaje.h"
```

Dependencia gráfica adjunta para msg_structures.h:



Este gráfico muestra que archivos directa o indirectamente incluyen a este archivo:



Clases

- struct msgnode_

  Nodo mensaje.

- struct msgvector_

  Vector de mensajes ya parseados.

- struct convnode_

  Nodo de los vectores de conversaciones.

Definiciones

- #define MQ_SIZE 20

Tipos definidos

- typedef msgnode_ msgnode

  Nodo mensaje.

- typedef msgvector_ msgvector

  Vector de mensajes ya parseados.

- typedef convnode_ convnode

  Nodo de los vectores de conversaciones.

Funciones

- void initmsglist ()

  Inicializa la cola de mensajes y las de conversaciones.

- int put_msg (Tmensaje *m, char *conv_id)
- void print_msg_queue ()

  Muestra el contenido de la cola de mensajes.

- void fprint_msg_queue ()

  Imprime el contenido de la cola de mensajes en un fichero.

- int set_active_conversation (int c)
- int set_terminated_conversation (int c)
- int set_pending_conversation (int c)
- Tmensaje * get_top_from_conv (int c)
- Tmensaje * get_last_from_conv (int c)
- int put_in_outbox (Tmensaje *msg)
- void print_outbox ()

  Imprime la bandeja de salida.

- void print_conv (char *conv_id)

Variables

- msgvector * msg_queue
- msgvector outbox

  Bandeja de salida de los mensaje.

4.12.1. Documentación de las definiciones

4.12.1.1. #define MQ_SIZE 20

Definición en la línea 9 del archivo msg_structures.h.

Referenciado por del_conv_from_terminated(), delete_conv_perm(), fprint_msg_queue(), hash_conv(), init_conv_list(), initmsglist(), is_active(), is_pending(), is_terminated(), print_conv(), print_msg_queue(), print_outbox(), remove_conversation_from_vector(), set_active_conversation(), set_pending_conversation(), set_terminated_conversation(), y updatefree().

4.12.2. Documentación de los tipos definidos

4.12.2.1. typedef struct convnode_ convnode

Nodo de los vectores de conversaciones.

4.12.2.2. typedef struct msgnode_ msgnode

Nodo mensaje.

Referenciado por como_send_erroneous().

4.12.2.3. typedef struct msgvector_ msgvector

Vector de mensajes ya parseados.

4.12.3. Documentación de las funciones

4.12.3.1. void fprint_msg_queue ()

Imprime el contenido de la cola de mensajes en un fichero.

Definición en la línea 399 del archivo msg_structures.c.

Hace referencia a aux, msgnode_::code, MQ_SIZE, msgnode_::msg, msg_queue, Tmensaje::performative, Tmensaje::receiver, Tmensaje::sender, TEXT_LENGTH, y msgvector_::v.

Referenciado por main().

4.12.3.2. Tmensaje* get_last_from_conv (int *c*)

Devuelve un puntero al ultimo mensaje de una conversacion

Parámetros:

c Indice de la conversacion en "msg_queue"

Devuelve:
El puntero al mensaje

Definición en la línea 709 del archivo msg_structures.c.

Hace referencia a msgvector_::last, msgnode_::msg, msg_queue, pm, y msgvector_::v.

4.12.3.3. Tmensaje* get_top_from_conv (int *c*)

Devuelve un puntero al primer mensaje efectivo de una conversacion

Parámetros:
c Indice de la conversacion en "msg_queue"

Devuelve:
El puntero al mensaje

Definición en la línea 698 del archivo msg_structures.c.

Hace referencia a msgnode_::msg, msg_queue, pm, msgvector_::top, y msgvector_::v.

Referenciado por como_receive().

4.12.3.4. void initmsglist ()

Inicializa la cola de mensajes y las de conversaciones.

Definición en la línea 53 del archivo msg_structures.c.

Hace referencia a active_convs, msgnode_::code, msgvector_::firstfree, init_conv_-list(), init_dict(), msgvector_::last, MQ_SIZE, msg_queue, names_table, outbox, pending_convs, terminated_convs, the_msg_queue, msgvector_::top, y msgvector_::v.

Referenciado por como_init(), y main().

4.12.3.5. void print_conv (char * *conv_id*)

Imprime una conversacion

Parámetros:
conv_id La conversacion

Definición en la línea 303 del archivo msg_structures.c.

Hace referencia a aux, msgnode_::code, debug_print(), get_conv_index_no_create(), MQ_SIZE, msgnode_::msg, msg_queue, Tmensaje::performative, Tmensaje::receiver, Tmensaje::sender, TEXT_LENGTH, y msgvector_::v.

Referenciado por processcommand().

4.12.3.6. void print_msg_queue ()

Muestra el contenido de la cola de mensajes.

Definición en la línea 334 del archivo msg_structures.c.

Hace referencia a active_convs, aux, msgnode_::code, debug_print(), MQ_-SIZE, msgnode_::msg, msg_queue, pending_convs, Tmensaje::performative, print_-outbox(), Tmensaje::receiver, Tmensaje::sender, terminated_convs, TEXT_LENGTH, y msgvector_::v.

Referenciado por main(), y processcommand().

4.12.3.7. void print_outbox ()

Imprime la bandeja de salida.

Definición en la línea 274 del archivo msg_structures.c.

Hace referencia a aux, msgnode_::code, Tmensaje::conversation_id, debug_-print(), msgvector_::firstfree, msgvector_::last, MQ_SIZE, msgnode_::msg, outbox, Tmensaje::performative, Tmensaje::receiver, Tmensaje::sender, TEXT_LENGTH, msgvector_::top, y msgvector_::v.

Referenciado por print_msg_queue(), y processcommand().

4.12.3.8. int put_in_outbox (Tmensaje * *msg*)

Pone un mensaje en la bandeja de salida

Parámetros:

msg El mensaje

Devuelve:

Codigo de error

Definición en la línea 768 del archivo msg_structures.c.

Hace referencia a aux, msgnode_::code, copymsgnode(), debug_print(), msgvector_-::firstfree, msgvector_::last, msgnode_::msg, outbox, TEXT_LENGTH, msgvector_-::top, updatefree(), y msgvector_::v.

Referenciado por como_send().

4.12.3.9. int put_msg (Tmensaje * *m*, char * *conv_id*)

Deposita un mensaje recien parseado en la cola

Parámetros:

m El mensaje

conv_id El identificador de conversacion

Devuelve:
Indice de la conversacion en 'msg_queue' o codigo de error (<0)

Definición en la línea 239 del archivo msg_structures.c.

Hace referencia a addmsgnode(), aux, debug_print(), get_conv_index(), is_terminated(), y TEXT_LENGTH.

Referenciado por como_receive_inbox(), main(), y mi_listen().

4.12.3.10. int set_active_conversation (int *c*)

Pone una conversacion en la lista de Conversaciones Activas

Parámetros:
c El indice de la conversacion en la cola de mensajes 'msg_queue'

Definición en la línea 546 del archivo msg_structures.c.

Hace referencia a active_convs, ERROR_NONE, MQ_SIZE, pending_convs, y remove_conversation_from_vector().

Referenciado por como_receive(), y como_send_outbox().

4.12.3.11. int set_pending_conversation (int *c*)

Pone una conversacion en la lista de Conversaciones Pendientes

Parámetros:
c El indice de la conversacion en la cola de mensajes 'msg_queue'

Definición en la línea 598 del archivo msg_structures.c.

Hace referencia a ERROR_NONE, MQ_SIZE, y pending_convs.

Referenciado por como_receive_inbox().

4.12.3.12. int set_terminated_conversation (int *c*)

Pone una conversacion en la lista de Conversaciones Terminadas

Parámetros:
c El indice de la conversacion en la cola de mensajes 'msg_queue'

Definición en la línea 572 del archivo msg_structures.c.

Hace referencia a active_convs, ERROR_NONE, MQ_SIZE, remove_conversation_from_vector(), y terminated_convs.

Referenciado por como_close_conversation().

4.12.4. Documentación de las variables

4.12.4.1. msgvector* msg_queue

Cola de conversaciones

Cola de conversaciones, compuesta de vectores de mensajes. La posicion 0 hace referencia a los mensajes erroneos

Definición en la línea 29 del archivo msg_structures.h.

Referenciado por addmsgnode(), como_send_erroneous(), delete_conv_perm(), deltopmsg(), fprint_msg_queue(), get_last_from_conv(), get_top_from_conv(), initmsglist(), print_conv(), print_msg_queue(), y updatefree().

4.12.4.2. msgvector outbox

Bandeja de salida de los mensaje.

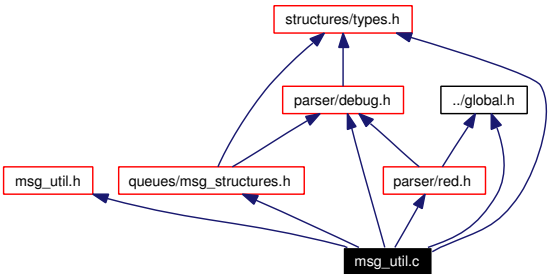
Definición en la línea 32 del archivo msg_structures.h.

Referenciado por como_send_outbox(), deltopmsg(), initmsglist(), print_outbox(), put_in_outbox(), y updatefree().

4.13. Referencia del Archivo msg_util.c

```
#include "msg_util.h"
#include "structures/types.h"
#include "parser/debug.h"
#include "parser/global.h"
#include "parser/red.h"
#include "queues/msg_structures.h"
```

Dependencia gráfica adjunta para msg_util.c:



Funciones

- int reply_to_error (Tmensaje *msg)

4.13.1. Documentación de las funciones

4.13.1.1. int reply_to_error (Tmensaje * *msg*)

Construye y envia una contestacion a un error

Construye un mensaje SIMBA de tipo "NOT UNDERSTOOD" en contestacion a un mensaje erroneo y lo envia sin pasar por la bandeja de salida. Esta funcion esta pensada para usarse como un mecanismo transparente de la comunicaciÃ179n

Parámetros:

msg Mensaje erroneo a contestar

Devuelve:

Error si lo hubiere

Definición en la línea 9 del archivo msg_util.c.

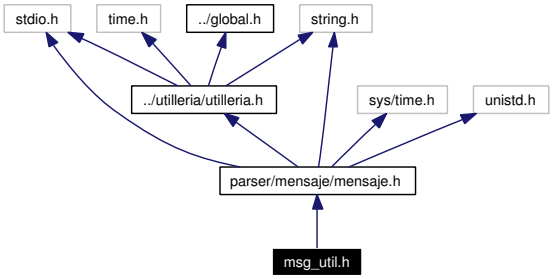
Hace referencia a Tmensaje::conversation_id, enviar(), Tmensaje::in_reply_to, Tmensaje::performative, Tmensaje::receiver, Tmensaje::reply_to, Tmensaje::reply_with, Tmensaje::sender, y Tmensaje::x_priority.

Referenciado por como_send_erroneous().

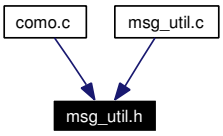
4.14. Referencia del Archivo msg_util.h

```
#include "parser/mensaje/mensaje.h"
```

Dependencia gráfica adjunta para msg_util.h:



Este gráfico muestra que archivos directa o indirectamente incluyen a este archivo:



Funciones

- int reply_to_error (Tmensaje *msg)

4.14.1. Documentación de las funciones

4.14.1.1. int reply_to_error (Tmensaje * *msg*)

Construye y envia una contestacion a un error

Construye un mensaje SIMBA de tipo "NOT UNDERSTOOD" en contestacion a un mensaje erroneo y lo envia sin pasar por la bandeja de salida. Esta funcion esta pensada para usarse como un mecanismo transparente de la comunicaciÃ179n

Parámetros:

msg Mensaje erroneo a contestar

Devuelve:

Error si lo hubiere

Definición en la línea 9 del archivo msg_util.c.

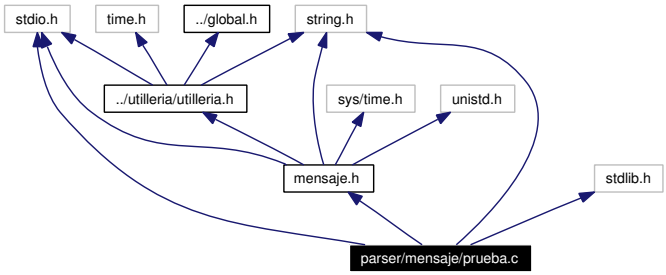
Hace referencia a Tmensaje::conversation_id, enviar(), Tmensaje::in_reply_to, Tmensaje::performative, Tmensaje::receiver, Tmensaje::reply_to, Tmensaje::reply_with, Tmensaje::sender, y Tmensaje::x_priority.

Referenciado por como_send_erroneous().

4.15. Referencia del Archivo prueba.c

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include "mensaje.h"
```

Dependencia gráfica adjunta para parser/mensaje/prueba.c:



Definiciones

- #define TAMLINEA 80
- #define TAMMSG 1000

Funciones

- int main (int argc, char *argv[ ])

Variables

- FILE * fd
- int nueva_entrada
- char linea [TAMLINEA]
- char mensaje [TAMMSG]
- Tmensaje campos

4.15.1. Documentación de las definiciones

4.15.1.1. #define TAMLINEA 80

Definición en la línea 7 del archivo parser/mensaje/prueba.c.

4.15.1.2. #define TAMMSG 1000

Definición en la línea 8 del archivo parser/mensaje/prueba.c.

4.15.2. Documentación de las funciones

4.15.2.1. int main (int *argc*, char * *argv*[ ])

Definición en la línea 24 del archivo parser/mensaje/prueba.c.

Hace referencia a campos, extraer_campos_mensaje(), fd, imprimir_campos_-mensaje(), linea, mensaje, y TAMLINEA.

4.15.3. Documentación de las variables

4.15.3.1. Tmensaje campos

Definición en la línea 22 del archivo parser/mensaje/prueba.c.

Referenciado por main().

4.15.3.2. FILE* fd

Definición en la línea 17 del archivo parser/mensaje/prueba.c.

Referenciado por main(), y read_msg_from_file().

4.15.3.3. char linea[TAMLINEA]

Definición en la línea 20 del archivo parser/mensaje/prueba.c.

Referenciado por main(), y read_msg_from_file().

4.15.3.4. char mensaje[TAMMSG]

Definición en la línea 21 del archivo parser/mensaje/prueba.c.

Referenciado por formar_mensaje(), main(), y read_msg_from_file().

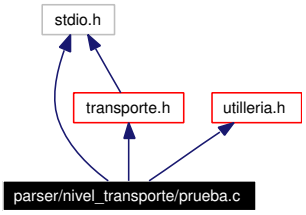
4.15.3.5. int nueva_entrada

Definición en la línea 19 del archivo parser/mensaje/prueba.c.

4.16. Referencia del Archivo prueba.c

```
#include <stdio.h>
#include "transporte.h"
#include "utilleria.h"
```

Dependencia gráfica adjunta para parser/nivel_transporte/prueba.c:



Funciones

- int main ()

4.16.1. Documentación de las funciones

4.16.1.1. int main ()

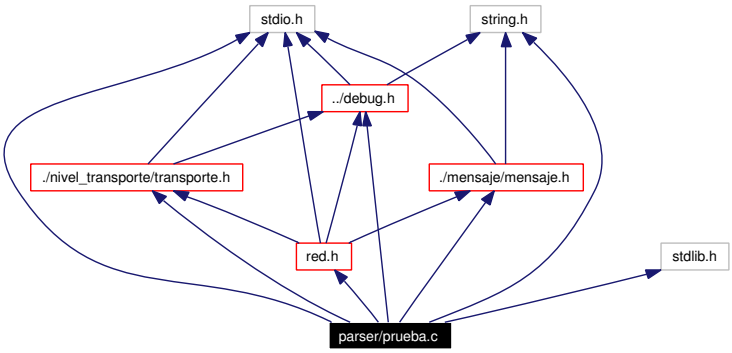
Definición en la línea 5 del archivo parser/nivel_transporte/prueba.c.

Hace referencia a enviar_udp(), itoa(), mensaje, presenta_socket_enviar(), y TAM_-MSG.

4.17. Referencia del Archivo prueba.c

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include "./mensaje/mensaje.h"
#include "./nivel_transporte/transporte.h"
#include "debug.h"
#include "red.h"
```

Dependencia gráfica adjunta para parser/prueba.c:



Definiciones

- #define TAMLINEA 255
- #define TAMMSG 10240

Funciones

- int main (int argc, char *argv[ ])

Variables

- FILE * fd
- int nueva_entrada
- char linea [TAMLINEA]
- char mensaje [TAMMSG]
- Tmensaje campos

4.17.1. Documentación de las definiciones

4.17.1.1. #define TAMLINEA 255

Definición en la línea 10 del archivo parser/prueba.c.

4.17.1.2. #define TAMMSG 10240

Definición en la línea 11 del archivo parser/prueba.c.

4.17.2. Documentación de las funciones

4.17.2.1. int main (int *argc*, char * *argv*[ ])

Definición en la línea 26 del archivo parser/prueba.c.

Hace referencia a campos, debug_print(), debug_show_log(), enviar(), extraer_campos_mensaje(), fd, linea, mensaje, y TAMLINEA.

4.17.3. Documentación de las variables

4.17.3.1. Tmensaje campos

Definición en la línea 24 del archivo parser/prueba.c.

4.17.3.2. FILE* fd

Definición en la línea 19 del archivo parser/prueba.c.

4.17.3.3. char linea[TAMLINEA]

Definición en la línea 22 del archivo parser/prueba.c.

4.17.3.4. char mensaje[TAMMSG]

Definición en la línea 23 del archivo parser/prueba.c.

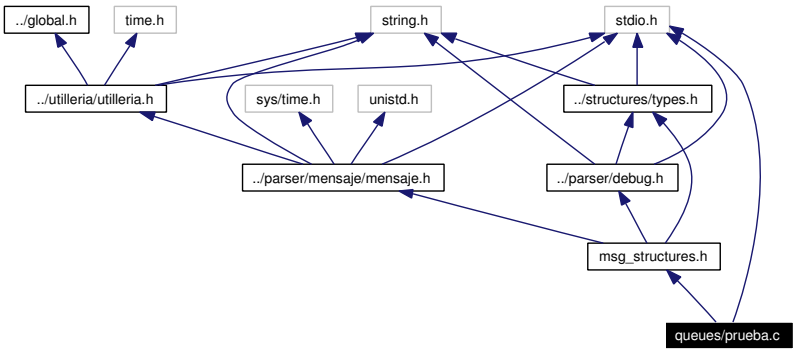
4.17.3.5. int nueva_entrada

Definición en la línea 21 del archivo parser/prueba.c.

4.18. Referencia del Archivo prueba.c

```
#include <stdio.h>
#include "msg_structures.h"
```

Dependencia gráfica adjunta para queues/prueba.c:



Funciones

- int main ()

4.18.1. Documentación de las funciones

4.18.1.1. int main ()

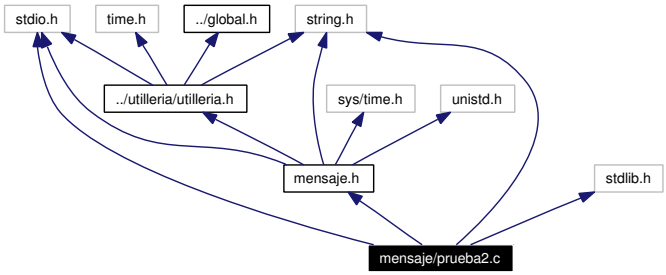
Definición en la línea 5 del archivo queues/prueba.c.

Hace referencia a Tmensaje::conversation_id, initmsglist(), Tmensaje::performative, pm, print_msg_queue(), put_msg(), Tmensaje::receiver, y Tmensaje::sender.

4.19. Referencia del Archivo prueba2.c

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include "mensaje.h"
```

Dependencia gráfica adjunta para mensaje/prueba2.c:



Definiciones

- #define TAMLINEA 80
- #define TAMMSG 1000

Funciones

- int main (int argc, char *argv[ ])

Variables

- FILE * fd
- int nueva_entrada
- char linea [TAMLINEA]
- char mensaje [TAMMSG]
- Tmensaje campos

4.19.1. Documentación de las definiciones

4.19.1.1. #define TAMLINEA 80

Definición en la línea 7 del archivo mensaje/prueba2.c.

4.19.1.2. #define TAMMSG 1000

Definición en la línea 8 del archivo mensaje/prueba2.c.

4.19.2. Documentación de las funciones

4.19.2.1. int main (int *argc*, char * *argv*[ ])

Definición en la línea 24 del archivo mensaje/prueba2.c.

Hace referencia a campos, extraer_campos_mensaje(), fd, formar_mensaje(), imprimir_campos_mensaje(), linea, mensaje, quitar_caracteres_inutiles(), y TAMLINEA.

4.19.3. Documentación de las variables

4.19.3.1. Tmensaje campos

Definición en la línea 22 del archivo mensaje/prueba2.c.

4.19.3.2. FILE* fd

Definición en la línea 17 del archivo mensaje/prueba2.c.

4.19.3.3. char linea[TAMLINEA]

Definición en la línea 20 del archivo mensaje/prueba2.c.

4.19.3.4. char mensaje[TAMMSG]

Definición en la línea 21 del archivo mensaje/prueba2.c.

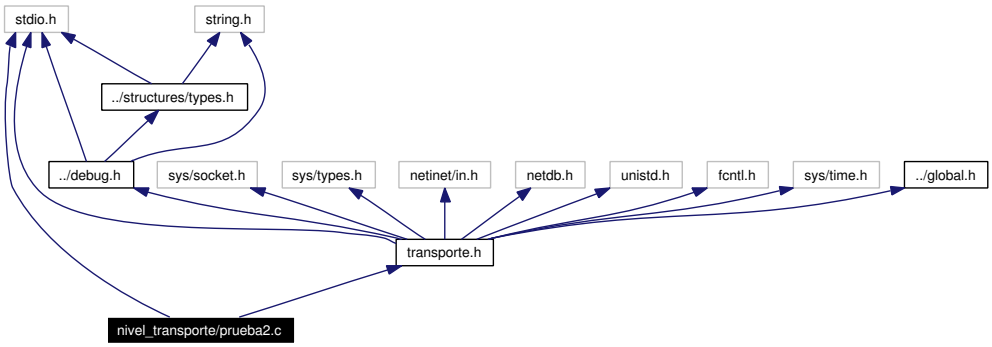
4.19.3.5. int nueva_entrada

Definición en la línea 19 del archivo mensaje/prueba2.c.

4.20. Referencia del Archivo prueba2.c

```
#include <stdio.h>
#include "transporte.h"
```

Dependencia gráfica adjunta para nivel_transporte/prueba2.c:



Funciones

- int main ()

4.20.1. Documentación de las funciones

4.20.1.1. int main ()

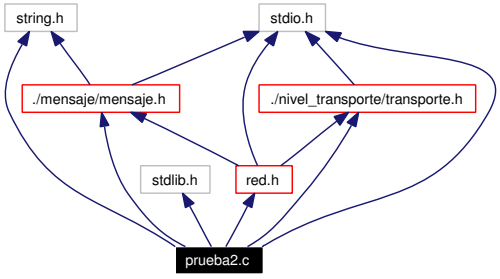
Definición en la línea 5 del archivo nivel_transporte/prueba2.c.

Hace referencia a inicializar_udp(), recibir_udp(), TAM_DIRECCION_IP, y TAM_MSG.

4.21. Referencia del Archivo prueba2.c

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include "./mensaje/mensaje.h"
#include "./nivel_transporte/transporte.h"
#include "red.h"
```

Dependencia gráfica adjunta para prueba2.c:



Definiciones

- #define TAMLINEA 80
- #define TAMMSG 1000

Funciones

- int main (int argc, char ∗argv[ ])

Variables

- FILE ∗ fd
- int nueva_entrada
- char linea [TAMLINEA]
- char mensaje [TAMMSG]
- char direccion_ip [TAMLINEA]
- Tmensaje campos
- char ∗ direccion

4.21.1. Documentación de las definiciones

4.21.1.1. #define TAMLINEA 80

Definición en la línea 9 del archivo prueba2.c.

4.21.1.2. #define TAMMSG 1000

Definición en la línea 10 del archivo prueba2.c.

4.21.2. Documentación de las funciones

4.21.2.1. int main (int *argc*, char * *argv*[ ])

Definición en la línea 27 del archivo prueba2.c.

Hace referencia a campos, direccion_ip, inicializar_red(), obtener_mi_direccion_ip(), y recibir().

4.21.3. Documentación de las variables

4.21.3.1. Tmensaje campos

Definición en la línea 25 del archivo prueba2.c.

4.21.3.2. char* direccion

Definición en la línea 26 del archivo prueba2.c.

Referenciado por enviar_udp(), inicializar_udp(), obtener_mi_direccion_ip(), presenta_socket_enviar(), y send_parted().

4.21.3.3. char direccion_ip[TAMLINEA]

Definición en la línea 24 del archivo prueba2.c.

Referenciado por como_init(), y main().

4.21.3.4. FILE* fd

Definición en la línea 19 del archivo prueba2.c.

4.21.3.5. char linea[TAMLINEA]

Definición en la línea 22 del archivo prueba2.c.

4.21.3.6. char mensaje[TAMMSG]

Definición en la línea 23 del archivo prueba2.c.

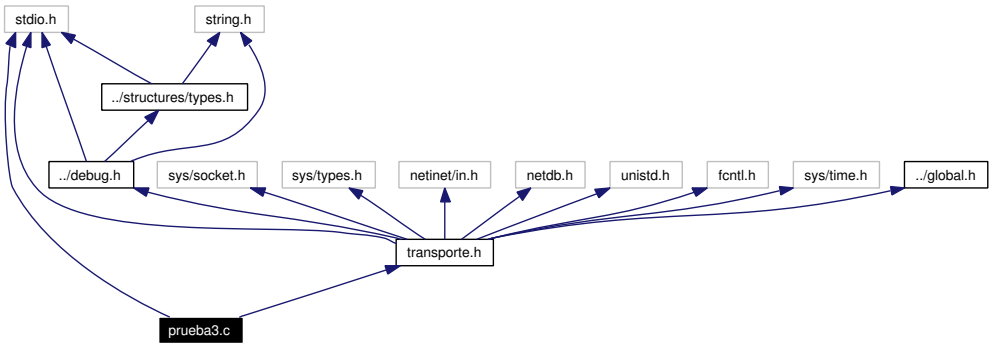
4.21.3.7. int nueva_entrada

Definición en la línea 21 del archivo prueba2.c.

4.22. Referencia del Archivo prueba3.c

```
#include <stdio.h>
#include "transporte.h"
```

Dependencia gráfica adjunta para prueba3.c:



Funciones

- int main ()

4.22.1. Documentación de las funciones

4.22.1.1. int main ()

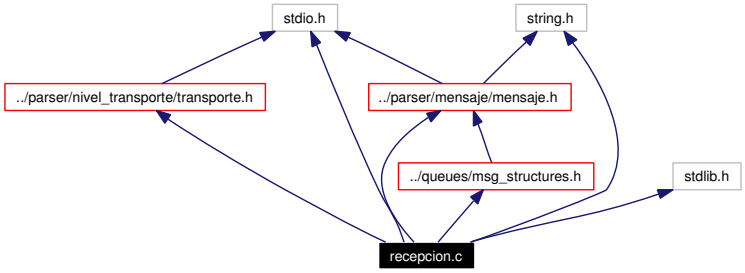
Definición en la línea 5 del archivo prueba3.c.

Hace referencia a formar_mensaje(), y TAM_MSG.

4.23. Referencia del Archivo recepcion.c

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include "../parser/mensaje/mensaje.h"
#include "../parser/nivel_transporte/transporte.h"
#include "../queues/msg_structures.h"
```

Dependencia gráfica adjunta para recepcion.c:



Definiciones

- #define TAMLINEA 255
- #define TAMMSG 10240

Funciones

- int main ()

Variables

- FILE * fd
- int nueva_entrada
- char linea [TAMLINEA]
- char mensaje [TAMMSG]
- char direccion_ip [TAMLINEA]
- Tmensaje campos
- char * direccion

4.23.1. Documentación de las definiciones

4.23.1.1. #define TAMLINEA 255

Definición en la línea 9 del archivo recepcion.c.

4.23.1.2. #define TAMMSG 10240

Definición en la línea 10 del archivo recepcion.c.

4.23.2. Documentación de las funciones

4.23.2.1. int main ()

Definición en la línea 23 del archivo recepcion.c.

Hace referencia a Tmensaje::conversation_id, direccion_ip, inicializar_red(), initmsglist(), obtener_mi_direccion_ip(), pm, print_msg_queue(), put_msg(), y recibir().

4.23.3. Documentación de las variables

4.23.3.1. Tmensaje campos

Definición en la línea 19 del archivo recepcion.c.

4.23.3.2. char* direccion

Definición en la línea 20 del archivo recepcion.c.

4.23.3.3. char direccion_ip[TAMLINEA]

Definición en la línea 18 del archivo recepcion.c.

4.23.3.4. FILE* fd

Definición en la línea 13 del archivo recepcion.c.

4.23.3.5. char linea[TAMLINEA]

Definición en la línea 16 del archivo recepcion.c.

4.23.3.6. char mensaje[TAMMSG]

Definición en la línea 17 del archivo recepcion.c.

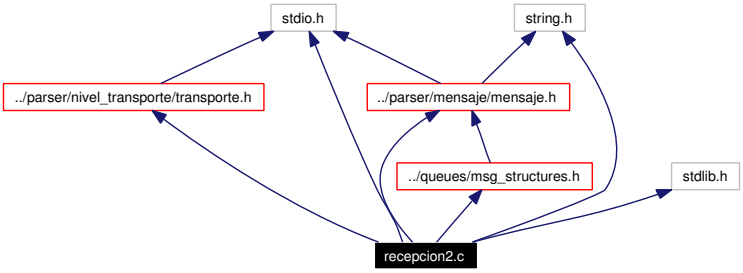
4.23.3.7. int nueva_entrada

Definición en la línea 15 del archivo recepcion.c.

4.24. Referencia del Archivo recepcion2.c

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include "../parser/mensaje/mensaje.h"
#include "../parser/nivel_transporte/transporte.h"
#include "../queues/msg_structures.h"
```

Dependencia gráfica adjunta para recepcion2.c:



Definiciones

- #define TAMLINEA 255
- #define TAMMSG 10240

Funciones

- int main ()

Variables

- FILE * fd
- int nueva_entrada
- char linea [TAMLINEA]
- char mensaje [TAMMSG]
- char direccion_ip [TAMLINEA]
- Tmensaje campos
- char * direccion

4.24.1. Documentación de las definiciones

4.24.1.1. #define TAMLINEA 255

Definición en la línea 9 del archivo recepcion2.c.

4.24.1.2. #define TAMMSG 10240

Definición en la línea 10 del archivo recepcion2.c.

4.24.2. Documentación de las funciones

4.24.2.1. int main ()

Definición en la línea 23 del archivo recepcion2.c.

Hace referencia a direccion_ip, fprint_msg_queue(), inicializar_red(), initmsglist(), obtener_mi_direccion_ip(), pm, put_msg(), y recibir().

4.24.3. Documentación de las variables

4.24.3.1. Tmensaje campos

Definición en la línea 19 del archivo recepcion2.c.

4.24.3.2. char* direccion

Definición en la línea 20 del archivo recepcion2.c.

4.24.3.3. char direccion_ip[TAMLINEA]

Definición en la línea 18 del archivo recepcion2.c.

4.24.3.4. FILE* fd

Definición en la línea 13 del archivo recepcion2.c.

4.24.3.5. char linea[TAMLINEA]

Definición en la línea 16 del archivo recepcion2.c.

4.24.3.6. char mensaje[TAMMSG]

Definición en la línea 17 del archivo recepcion2.c.

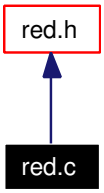
4.24.3.7. int nueva_entrada

Definición en la línea 15 del archivo recepcion2.c.

4.25. Referencia del Archivo red.c

```
#include "red.h"
```

Dependencia gráfica adjunta para red.c:



Funciones

- int enviar (Tmensaje campos_mensaje)
- int recibir (Tmensaje *campos_mensaje)
- int inicializar_red ()
- void obtener_mi_direccion_ip (char *direccion)

4.25.1. Documentación de las funciones

4.25.1.1. int enviar (Tmensaje *campos_mensaje*)

Enviar un mensaje

Funcion que se encarga de enviar un mensaje (msg) a traves de un socket conectado con destino. Si el socket no existe lo crea.

Parámetros:

campos_mensaje Mensaje a enviar

Devuelve:

Numero de caracteres enviados

Definición en la línea 5 del archivo red.c.

Hace referencia a debug_print(), enviar_udp(), formar_mensaje(), Tmensaje::receiver, y TAM_MSG.

Referenciado por como_send_outbox(), main(), y reply_to_error().

4.25.1.2. int inicializar_red ()

Inicializar red

Funcion que recibe datos a traves de un socket

Definición en la línea 64 del archivo red.c.

Hace referencia a inicializar_udp().

Referenciado por como_init(), y main().

4.25.1.3. void obtener_mi_direccion_ip (char * *direccion*)

Devuelve la propia direccion IP

Devuelve la direccion ip de la maquina sobre la que esta oyendo el servidor

Parámetros:

direccion Cadena pasada por referencia donde almacenar la direccion obtenida

Definición en la línea 73 del archivo red.c.

Hace referencia a debug_print(), direccion, obtener_dir_ip_servidor(), y TAM_MSG.

Referenciado por como_init(), y main().

4.25.1.4. int recibir (Tmensaje * *campos_mensaje*)

Recibir datos

Funcion que recibe datos a traves de un socket Comprueba si esta conectado. En caso contrario lo conecta. Lee del socket ya conectado

Parámetros:

campos_mensaje Mensaje a recibir

Devuelve:

Numero de caracteres leidos

Definición en la línea 33 del archivo red.c.

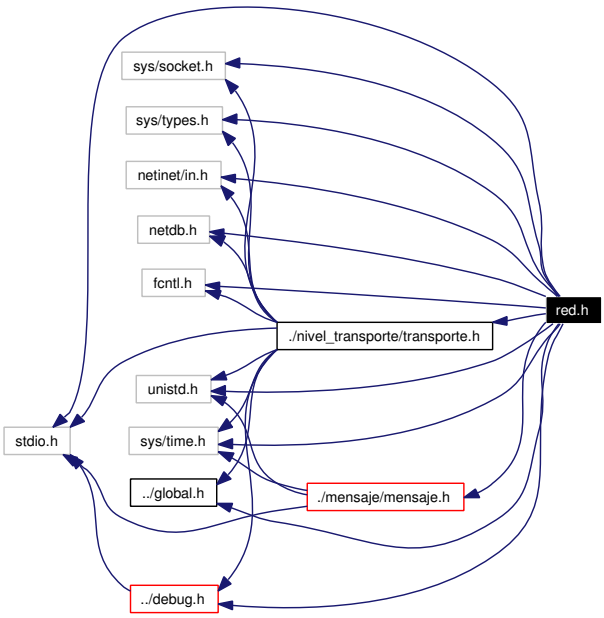
Hace referencia a debug_print(), extraer_campos_mensaje(), imprimir_campos_-mensaje(), obtener_dir_ip_servidor(), recibir_udp(), Tmensaje::sender, TAM_-DIRECCION_IP, y TAM_MSG.

Referenciado por como_receive_inbox(), main(), y mi_listen().

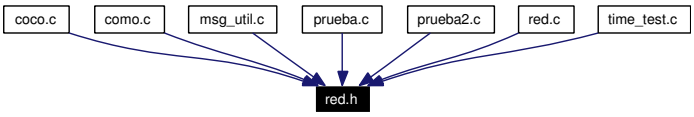
4.26. Referencia del Archivo red.h

```
#include <stdio.h>
#include <sys/socket.h>
#include <sys/types.h>
#include <netinet/in.h>
#include <netdb.h>
#include <unistd.h>
#include <fcntl.h>
#include <sys/time.h>
#include "./nivel_transporte/transporte.h"
#include "./mensaje/mensaje.h"
#include "global.h"
#include "debug.h"
```

Dependencia gráfica adjunta para red.h:



Este gráfico muestra que archivos directa o indirectamente incluyen a este archivo:



Funciones

- int enviar (Tmensaje campos_mensaje)
- int recibir (Tmensaje *campos_mensaje)
- int inicializar_red ()
- void obtener_mi_direccion_ip (char *direccion)

4.26.1. Documentación de las funciones

4.26.1.1. int enviar (Tmensaje *campos_mensaje*)

Enviar un mensaje

Funcion que se encarga de enviar un mensaje (msg) a traves de un socket conectado con destino. Si el socket no existe lo crea.

Parámetros:

campos_mensaje Mensaje a enviar

Devuelve:

Numero de caracteres enviados

Definición en la línea 5 del archivo red.c.

Hace referencia a debug_print(), enviar_udp(), formar_mensaje(), Tmensaje::receiver, y TAM_MSG.

Referenciado por como_send_outbox(), main(), y reply_to_error().

4.26.1.2. int inicializar_red ()

Inicializar red

Funcion que recibe datos a traves de un socket

Definición en la línea 64 del archivo red.c.

Hace referencia a inicializar_udp().

Referenciado por como_init(), y main().

4.26.1.3. void obtener_mi_direccion_ip (char * *direccion*)

Devuelve la propia direccion IP

Devuelve la direccion ip de la maquina sobre la que esta oyendo el servidor

Parámetros:

direccion Cadena pasada por referencia donde almacenar la direccion obtenida

Definición en la línea 73 del archivo red.c.

Hace referencia a debug_print(), direccion, obtener_dir_ip_servidor(), y TAM_MSG.

Referenciado por como_init(), y main().

4.26.1.4. int recibir (Tmensaje * *campos_mensaje*)

Recibir datos

Funcion que recibe datos a traves de un socket Comprueba si esta conectado. En caso contrario lo conecta. Lee del socket ya conectado

Parámetros:

campos_mensaje Mensaje a recibir

Devuelve:

Numero de caracteres leidos

Definición en la línea 33 del archivo red.c.

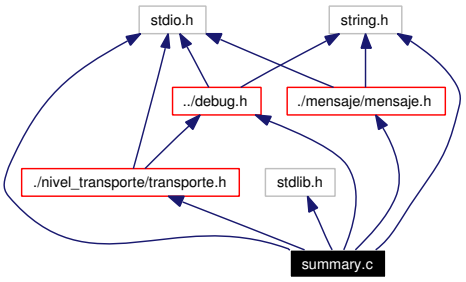
Hace referencia a debug_print(), extraer_campos_mensaje(), imprimir_campos_-mensaje(), obtener_dir_ip_servidor(), recibir_udp(), Tmensaje::sender, TAM_-DIRECCION_IP, y TAM_MSG.

Referenciado por como_receive_inbox(), main(), y mi_listen().

4.27. Referencia del Archivo summary.c

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include "./mensaje/mensaje.h"
#include "./nivel_transporte/transporte.h"
#include "debug.h"
```

Dependencia gráfica adjunta para summary.c:



Definiciones

- #define TAMLINEA 255
- #define TAMMSG 10240

Funciones

- int main (int argc, char *argv[ ])

Variables

- FILE * fd
- int nueva_entrada
- char linea [TAMLINEA]
- char mensaje [TAMMSG]

4.27.1. Documentación de las definiciones

4.27.1.1. #define TAMLINEA 255

Definición en la línea 9 del archivo summary.c.

4.27.1.2. #define TAMMSG 10240

Definición en la línea 10 del archivo summary.c.

4.27.2. Documentación de las funciones

4.27.2.1. int main (int *argc*, char * *argv*[ ])

Definición en la línea 17 del archivo summary.c.

Hace referencia a aux, debug_print(), fd, linea, mensaje, y TAMLINEA.

4.27.3. Documentación de las variables

4.27.3.1. FILE* fd

Definición en la línea 12 del archivo summary.c.

4.27.3.2. char linea[TAMLINEA]

Definición en la línea 14 del archivo summary.c.

4.27.3.3. char mensaje[TAMMSG]

Definición en la línea 15 del archivo summary.c.

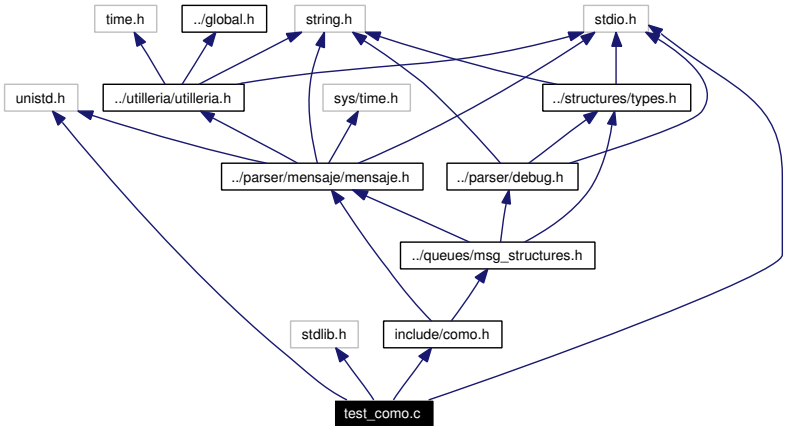
4.27.3.4. int nueva_entrada

Definición en la línea 13 del archivo summary.c.

4.28. Referencia del Archivo test_como.c

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include "include/como.h"
```

Dependencia gráfica adjunta para test_como.c:



Definiciones

- #define TAMLINEA 255
- #define TAMMSG 10240

Funciones

- int main (int argc, char *argv[ ])

Variables

- FILE * fd
- int nueva_entrada
- char linea [TAMLINEA]
- char mensaje [TAMMSG]

4.28.1. Documentación de las definiciones

4.28.1.1. #define TAMLINEA 255

Definición en la línea 7 del archivo test_como.c.

4.28.1.2. #define TAMMSG 10240

Definición en la línea 8 del archivo test_como.c.

4.28.2. Documentación de las funciones

4.28.2.1. int main (int *argc*, char * *argv*[ ])

Definición en la línea 17 del archivo test_como.c.

Hace referencia a como_init(), como_send(), fd, linea, mensaje, y TAMLINEA.

4.28.3. Documentación de las variables

4.28.3.1. FILE* fd

Definición en la línea 10 del archivo test_como.c.

4.28.3.2. char linea[TAMLINEA]

Definición en la línea 13 del archivo test_como.c.

4.28.3.3. char mensaje[TAMMSG]

Definición en la línea 14 del archivo test_como.c.

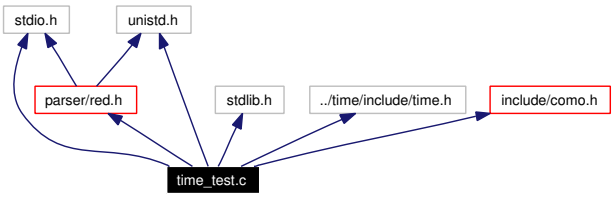
4.28.3.4. int nueva_entrada

Definición en la línea 12 del archivo test_como.c.

4.29. Referencia del Archivo time_test.c

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include "../time/include/time.h"
#include "parser/red.h"
#include "include/como.h"
```

Dependencia gráfica adjunta para time_test.c:



Definiciones

- #define TAMLINEA 255
- #define TAMMSG 10240

Funciones

- int read_msg_from_file (char *arg1, char *mensaje)
- int main (int argc, char *argv[ ])

4.29.1. Documentación de las definiciones

4.29.1.1. #define TAMLINEA 255

Definición en la línea 10 del archivo time_test.c.

4.29.1.2. #define TAMMSG 10240

Definición en la línea 11 del archivo time_test.c.

4.29.2. Documentación de las funciones

4.29.2.1. int main (int *argc*, char * *argv*[ ])

Definición en la línea 30 del archivo time_test.c.

Hace referencia a como_init(), como_send(), read_msg_from_file(), y TAMMSG.

4.29.2.2. int read_msg_from_file (char * *arg1*, char * *mensaje*)

Definición en la línea 13 del archivo time_test.c.

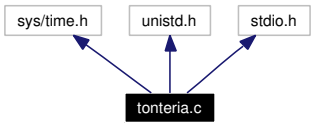
Hace referencia a fd, linea, mensaje, y TAMLINEA.

Referenciado por main(), y processcommand().

4.30. Referencia del Archivo tonteria.c

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

Dependencia gráfica adjunta para tonteria.c:



Funciones

- int main ()

Variables

- int pepe

4.30.1. Documentación de las funciones

4.30.1.1. int main ()

Definición en la línea 8 del archivo tonteria.c.

4.30.2. Documentación de las variables

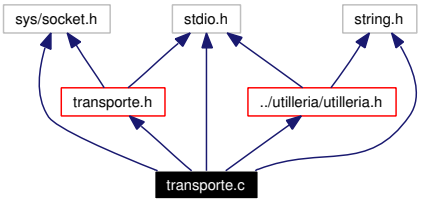
4.30.2.1. int pepe

Definición en la línea 5 del archivo tonteria.c.

4.31. Referencia del Archivo transporte.c

```
#include "transporte.h"
#include "../utilleria/utilleria.h"
#include <stdio.h>
#include <string.h>
#include <sys/socket.h>
```

Dependencia gráfica adjunta para transporte.c:



Clases

- struct tipo_cliente

Funciones

- long summarize (char *data)
- unsigned long int inet_addr (const char *cp)
- char * inet_ntoa (struct in_addr in)
- int summarize_chunk (unsigned char *partdata)
- int connectsock (char *host)
- void serialize_chunk (chunk *part, unsigned char *spart)
- int enviar_udp (char *msg, char *destino)

  Vector de buffers de recepcion de mensajes de bajo nivel.

- int recibir_udp (char msg[ ], char dir_cliente[ ])
- int crear_socket_escucha (void)
- void presenta_socket_enviar (void)

  Presenta los sockets de envio.

- char * obtener_dir_ip_servidor ()
- int escucha_multicast (int s)
- int inicializar_udp ()
- int send_parted (char *msg, char *dest)
- void clean_rcv_buffer (int index_client)
- int receive_parted (char msg[ ], char dir_cliente[ ])

Variables

- tipo_cliente cliente [MAX_ROBOTS]

  Vector de estructuras que guardara la informacion de los sockets conectados.

- int escucha

  Variable que contiene el socket de escucha del servidor.

- char dir_ip_escucha [TAM_DIRECCION_IP]

  Direccion Ip sobre la que esta ejecutandose el servidor.

- buffer_msg rcv_buffer [MAX_ROBOTS]

4.31.1. Documentación de las funciones

4.31.1.1. void clean_rcv_buffer (int *index_client*)

Definición en la línea 444 del archivo transporte.c.

Hace referencia a MSG_MAX_SIZE, y rcv_buffer.

Referenciado por receive_parted().

4.31.1.2. int connectsock (char * *host*)

Conecta a un socket remoto

Funcion que crea y realiza la conexion con un socket remoto para enviar informacion.

Parámetros:

host El host remoto

Devuelve:

El socket creado y ya conectado

Definición en la línea 89 del archivo transporte.c.

Hace referencia a aux, debug_print(), DIR_IP_GRUPO_MULTICAST, inet_addr(), SOCKET_RECIBIR, y TEXT_LENGTH.

Referenciado por enviar_udp(), inicializar_udp(), y send_parted().

4.31.1.3. int crear_socket_escucha (void)

Crea un socket de escucha

Funcion que crea un socket para la recepcion de mensajes

Devuelve:

el socket creado y conectado, -1 en caso de error

Definición en la línea 172 del archivo transporte.c.

Hace referencia a debug_print(), y SOCKET_RECIBIR.

Referenciado por inicializar_udp().

4.31.1.4. int enviar_udp (char * *msg*, char * *dst*)

Vector de buffers de recepcion de mensajes de bajo nivel.

Envia un mensaje a traves de un socket

Funcion que envia un mensaje al destino especificado a traves de un socket.

Parámetros:

msg El mensaje a enviar

dst Destino

Devuelve:

Numero de caracteres realmente enviados

Definición en la línea 34 del archivo transporte.c.

Hace referencia a aux, cliente, connectsock(), debug_print(), direccion, FALSE, ltoa(), MAX_ROBOTS, tipo_cliente::socket_cliente, summarize(), TEXT_LENGTH, y TRUE.

Referenciado por enviar(), y main().

4.31.1.5. int escucha_multicast (int *s*)

Pone a un socket a escuchar en modo multicast

Parámetros:

s El socket

Devuelve:

0 si todo ha ido bien, -1 en caso contrario

Definición en la línea 253 del archivo transporte.c.

Hace referencia a debug_print(), DIR_IP_GRUPO_MULTICAST, inet_addr(), y obtener_dir_ip_servidor().

Referenciado por inicializar_udp().

4.31.1.6. unsigned long int inet_addr (const char * *cp*)

Referenciado por connectsock(), y escucha_multicast().

4.31.1.7. char* inet_ntoa (struct in_addr *in*)

Referenciado por obtener_dir_ip_servidor(), receive_parted(), y recibir_udp().

4.31.1.8. int inicializar_udp ()

Inicializa la red

Funcion que inicializa el servicio de red por UDP

Definición en la línea 281 del archivo transporte.c.

Hace referencia a cliente, connectsock(), crear_socket_escucha(), debug_print(), dir_ip_escucha, DIR_IP_GRUPO_MULTICAST, direccion, escucha, escucha_multicast(), MAX_ROBOTS, MSG_MAX_SIZE, obtener_dir_ip_servidor(), rcv_buffer, y tipo_cliente::socket_cliente.

Referenciado por inicializar_red(), y main().

4.31.1.9. char* obtener_dir_ip_servidor ()

Obtiene la direccion IP del servidor

Funcion que obtiene la direccion IP del servidor

Devuelve:
Una cadena con la direccion IP principal del servidor

Definición en la línea 234 del archivo transporte.c.

Hace referencia a inet_ntoa(), y TAM_DIRECCION_IP.

Referenciado por escucha_multicast(), inicializar_udp(), obtener_mi_direccion_ip(), y recibir().

4.31.1.10. void presenta_socket_enviar (void)

Presenta los sockets de envio.

Definición en la línea 216 del archivo transporte.c.

Hace referencia a aux, cliente, debug_print(), direccion, itoa(), MAX_ROBOTS, y TEXT_LENGTH.

Referenciado por main().

4.31.1.11. int receive_parted (char *msg*[ ], char *dir_cliente*[ ])

Definición en la línea 452 del archivo transporte.c.

Hace referencia a CHUNK_SIZE, clean_rcv_buffer(), cliente, debug_print(), ERROR_NONE, escucha, inet_ntoa(), MAX_ROBOTS, rcv_buffer, summarize(), summarize_chunk(), y TEXT_LENGTH.

4.31.1.12. int recibir_udp (char *msg*[ ], char *dir_cliente*[ ])

Definición en la línea 140 del archivo transporte.c.

Hace referencia a cliente, escucha, inet_ntoa(), y TAM_MSG.

4.31.1.13. int send_parted (char * *msg*, char * *dest*)

Definición en la línea 331 del archivo transporte.c.

Hace referencia a aux, CHUNK_SIZE, cliente, connectsock(), debug_print(), direccion, FALSE, MAX_ROBOTS, tipo_cliente::socket_cliente, summarize_chunk(), TEXT_LENGTH, y TRUE.

4.31.1.14. void serialize_chunk (chunk * *part*, unsigned char * *spart*)

Definición en la línea 322 del archivo transporte.c.

4.31.1.15. long summarize (char * *data*)

Definición en la línea 38 del archivo nivel_transporte/utilleria.c.

4.31.1.16. int summarize_chunk (unsigned char * *partdata*)

Definición en la línea 49 del archivo nivel_transporte/utilleria.c.

4.31.2. Documentación de las variables

4.31.2.1. tipo_cliente cliente[MAX_ROBOTS] [static]

Vector de estructuras que guardara la informacion de los sockets conectados.

Definición en la línea 23 del archivo transporte.c.

Referenciado por enviar_udp(), inicializar_udp(), presenta_socket_enviar(), receive_parted(), recibir_udp(), y send_parted().

4.31.2.2. char dir_ip_escucha[TAM_DIRECCION_IP] [static]

Direccion Ip sobre la que esta ejecutandose el servidor.

Definición en la línea 29 del archivo transporte.c.

Referenciado por inicializar_udp().

4.31.2.3. int escucha [static]

Variable que contiene el socket de escucha del servidor.

Definición en la línea 26 del archivo transporte.c.

Referenciado por inicializar_udp(), receive_parted(), y recibir_udp().

4.31.2.4. buffer_msg rcv_buffer[MAX_ROBOTS] [static]

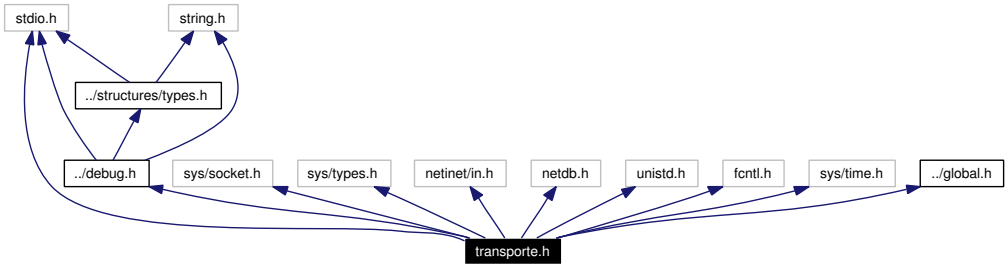
Definición en la línea 31 del archivo transporte.c.

Referenciado por clean_rcv_buffer(), inicializar_udp(), y receive_parted().

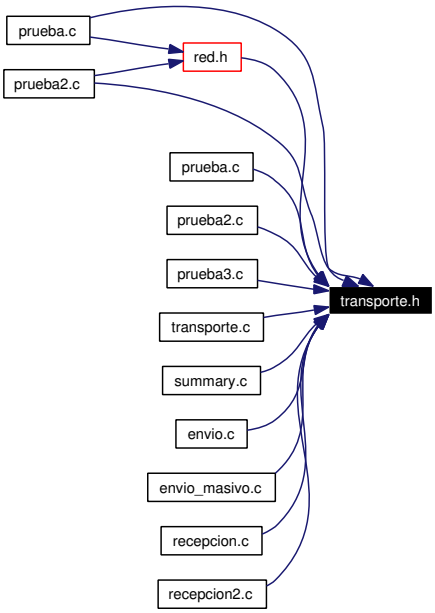
4.32. Referencia del Archivo transporte.h

```
#include <stdio.h>
#include <sys/socket.h>
#include <sys/types.h>
#include <netinet/in.h>
#include <netdb.h>
#include <unistd.h>
#include <fcntl.h>
#include <sys/time.h>
#include "../global.h"
#include "../debug.h"
```

Dependencia gráfica adjunta para transporte.h:



Este gráfico muestra que archivos directa o indirectamente incluyen a este archivo:



Definiciones

- #define SOCKET_RECIBIR 2001
- #define DIR_IP_GRUPO_MULTICAST "225.0.0.1"

Tipos definidos

- typedef char buffer_msg [MSG_MAX_SIZE]

Funciones

- int enviar_udp (char *msg, char *dst)

  Vector de buffers de recepcion de mensajes de bajo nivel.

- int recibir_udp (char *msg, char *dir_cliente)
- int inicializar_udp ()
- char * obtener_dir_ip_servidor ()

4.32.1. Documentación de las definiciones

4.32.1.1. #define DIR_IP_GRUPO_MULTICAST "225.0.0.1"

Definición en la línea 22 del archivo transporte.h.

Referenciado por connectsock(), escucha_multicast(), y inicializar_udp().

4.32.1.2. #define SOCKET_RECIBIR 2001

Definición en la línea 18 del archivo transporte.h.

Referenciado por connectsock(), y crear_socket_escucha().

4.32.2. Documentación de los tipos definidos

4.32.2.1. typedef char buffer_msg[MSG_MAX_SIZE]

Definición en la línea 24 del archivo transporte.h.

4.32.3. Documentación de las funciones

4.32.3.1. int enviar_udp (char * *msg*, char * *dst*)

Vector de buffers de recepcion de mensajes de bajo nivel.

Envia un mensaje a traves de un socket

Funcion que envia un mensaje al destino especificado a traves de un socket.

Parámetros:

msg El mensaje a enviar

dst Destino

Devuelve:

Numero de caracteres realmente enviados

Definición en la línea 34 del archivo transporte.c.

Hace referencia a aux, cliente, connectsock(), debug_print(), direccion, FALSE, ltoa(), MAX_ROBOTS, tipo_cliente::socket_cliente, summarize(), TEXT_LENGTH, y TRUE.

Referenciado por enviar(), y main().

4.32.3.2. int inicializar_udp ()

Inicializa la red

Funcion que inicializa el servicio de red por UDP

Definición en la línea 281 del archivo transporte.c.

Hace referencia a cliente, connectsock(), crear_socket_escucha(), debug_print(), dir_ip_escucha, DIR_IP_GRUPO_MULTICAST, direccion, escucha, escucha_multicast(), MAX_ROBOTS, MSG_MAX_SIZE, obtener_dir_ip_servidor(), rcv_buffer, y tipo_cliente::socket_cliente.

Referenciado por inicializar_red(), y main().

4.32.3.3. char* obtener_dir_ip_servidor ()

Obtiene la direccion IP del servidor

Funcion que obtiene la direccion IP del servidor

Devuelve:
Una cadena con la direccion IP principal del servidor

Definición en la línea 234 del archivo transporte.c.

Hace referencia a inet_ntoa(), y TAM_DIRECCION_IP.

Referenciado por escucha_multicast(), inicializar_udp(), obtener_mi_direccion_ip(), y recibir().

4.32.3.4. int recibir_udp (char * *msg*, char * *dir_cliente*)

Recibe un mensaje

Funcion que recibe un mensaje a traves de un socket.

Parámetros:
msg El mensaje a recibir

dir_cliente Direccion del cliente

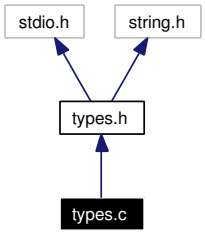
Devuelve:
Numero de caracteres realmente leidos

Referenciado por main(), y recibir().

4.33. Referencia del Archivo types.c

```
#include "types.h"
```

Dependencia gráfica adjunta para types.c:



Funciones

- int initlist (linkedlist l)
- void printlist (linkedlist l)
- int addnode (linkedlist l)
- int setcode (linkedlist l, int index, int c)
- int setmsg (linkedlist l, int index, char m[TEXT_LENGTH])
- int len (linkedlist l)
- int getcode (linkedlist l, int index)
- void getmsg (linkedlist l, int index, char receiver[TEXT_LENGTH])
- int copynode (linkedlist l, int dest, int orig)

4.33.1. Documentación de las funciones

4.33.1.1. int addnode (linkedlist *l*)

Devuelve el indice del siguiente nodo libre de la lista

Parámetros:

l Lista enlazada

Definición en la línea 38 del archivo types.c.

Hace referencia a lnode_::code, y MAX_VECTOR.

Referenciado por debug_start(), y main().

4.33.1.2. int copynode (linkedlist *l*, int *dest*, int *orig*)

Copia un nodo a otra posicion de la misma lista

Parámetros:

l Lista enlazada

dest Indice del nodo destino

orig Indice del nodo origen

Definición en la línea 69 del archivo types.c.

Hace referencia a lnode_::code, y ERROR_NONE.

4.33.1.3. int getcode (linkedlist *l*, int *index*)

Obtiene el codigo de un nodo

Parámetros:

l Lista enlazada

index Indice del nodo

Devuelve:

El codigo del nodo

Definición en la línea 61 del archivo types.c.

Hace referencia a lnode_::code.

Referenciado por debug_show_log().

4.33.1.4. void getmsg (linkedlist *l*, int *index*, char *receiver*[TEXT_LENGTH])

Obtiene el mensaje de un nodo

Parámetros:

l Lista enlazada

index Indice del nodo

receiver Cadena de texto donde almacenar el mensaje

Definición en la línea 65 del archivo types.c.

Referenciado por debug_show_log().

4.33.1.5. int initlist (linkedlist *l*)

Inicializa una lista

Parámetros:

l Lista por iniciar

Definición en la línea 4 del archivo types.c.

Hace referencia a lnode_::code, ERROR_NONE, y MAX_VECTOR.

Referenciado por debug_start(), y main().

4.33.1.6. int len (linkedlist *l*)

Obtiene la longitud de una lista

Parámetros:
l Lista enlazada

Devuelve:
La longitud

Definición en la línea 57 del archivo types.c.

Hace referencia a lnode_::code.

Referenciado por debug_show_log().

4.33.1.7. void printlist (linkedlist *l*)

Imprime una lista

Parámetros:
l Lista a imprimir

Definición en la línea 20 del archivo types.c.

Hace referencia a lnode_::code.

Referenciado por main().

4.33.1.8. int setcode (linkedlist *l*, int *index*, int *c*)

Fija un codigo en un nodo

Parámetros:
l Lista enlazada
index Indice del nodo
c Codigo deseado

Definición en la línea 45 del archivo types.c.

Hace referencia a lnode_::code, y ERROR_NONE.

Referenciado por debug_start(), y main().

4.33.1.9. int setmsg (linkedlist *l*, int *index*, char *m*[TEXT_LENGTH])

Fija un mensaje en un nodo

Parámetros:
l Lista enlazada

index Indice del nodo

m Mensaje deseado

Definición en la línea 51 del archivo types.c.

Hace referencia a ERROR_NONE.

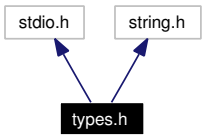
Referenciado por debug_start(), y main().

4.34. Referencia del Archivo types.h

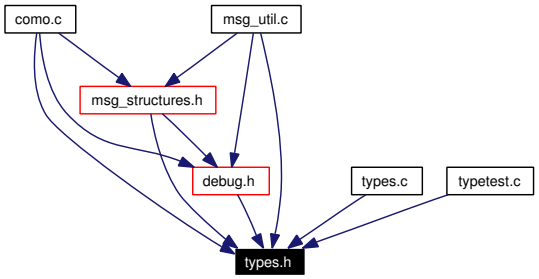
`#include <stdio.h>`

`#include <string.h>`

Dependencia gráfica adjunta para types.h:



Este gráfico muestra que archivos directa o indirectamente incluyen a este archivo:



Clases

- struct lnode_

  Nodo de una lista enlazada.

Definiciones

- #define MAX_VECTOR 255
- #define TEXT_LENGTH 1024
- #define LIST_LENGTH 1024
- #define ERROR_NONE 0

Tipos definidos

- typedef lnode_ lnode

  Nodo de una lista enlazada.

- typedef lnode linkedlist [MAX_VECTOR]

  Lista enlazada estatica.

Funciones

- int initlist (linkedlist l)
- void printlist (linkedlist l)
- int addnode (linkedlist l)
- int setcode (linkedlist l, int index, int c)
- int setmsg (linkedlist l, int index, char m[TEXT_LENGTH])
- int getcode (linkedlist l, int index)
- void getmsg (linkedlist l, int index, char receiver[TEXT_LENGTH])
- int len (linkedlist l)
- int copynode (linkedlist l, int dest, int orig)

4.34.1. Documentación de las definiciones

4.34.1.1. #define ERROR_NONE 0

Definición en la línea 17 del archivo types.h.

Referenciado por como_send(), copynode(), initlist(), quita_tiempo(), receive_parted(), set_active_conversation(), set_pending_conversation(), set_terminated_conversation(), setcode(), y setmsg().

4.34.1.2. #define LIST_LENGTH 1024

Definición en la línea 16 del archivo types.h.

4.34.1.3. #define MAX_VECTOR 255

Definición en la línea 14 del archivo types.h.

Referenciado por addnode(), y initlist().

4.34.1.4. #define TEXT_LENGTH 1024

Definición en la línea 15 del archivo types.h.

Referenciado por addmsgnode(), connectsock(), debug_show_log(), del_key(), enviar_udp(), extraer_campos_mensaje(), fprint_msg_queue(), presenta_socket_-enviar(), print_conv(), print_msg_queue(), print_outbox(), put_in_outbox(), put_-msg(), receive_parted(), send_parted(), y updatefree().

4.34.2. Documentación de los tipos definidos

4.34.2.1. typedef lnode linkedlist[MAX_VECTOR]

Lista enlazada estatica.

Definición en la línea 26 del archivo types.h.

4.34.2.2. typedef struct lnode_ lnode

Nodo de una lista enlazada.

4.34.3. Documentación de las funciones

4.34.3.1. int addnode (linkedlist *l*)

Devuelve el indice del siguiente nodo libre de la lista

Parámetros:

l Lista enlazada

Definición en la línea 38 del archivo types.c.

Hace referencia a lnode_::code, y MAX_VECTOR.

Referenciado por debug_start(), y main().

4.34.3.2. int copynode (linkedlist *l*, int *dest*, int *orig*)

Copia un nodo a otra posicion de la misma lista

Parámetros:

l Lista enlazada

dest Indice del nodo destino

orig Indice del nodo origen

Definición en la línea 69 del archivo types.c.

Hace referencia a lnode_::code, y ERROR_NONE.

4.34.3.3. int getcode (linkedlist *l*, int *index*)

Obtiene el codigo de un nodo

Parámetros:

l Lista enlazada

index Indice del nodo

Devuelve:

El codigo del nodo

Definición en la línea 61 del archivo types.c.

Hace referencia a lnode_::code.

Referenciado por debug_show_log().

4.34.3.4. void getmsg (linkedlist *l*, int *index*, char *receiver*[TEXT_LENGTH])

Obtiene el mensaje de un nodo

Parámetros:

l Lista enlazada

index Indice del nodo

receiver Cadena de texto donde almacenar el mensaje

Definición en la línea 65 del archivo types.c.

Referenciado por debug_show_log().

4.34.3.5. int initlist (linkedlist *l*)

Inicializa una lista

Parámetros:

l Lista por iniciar

Definición en la línea 4 del archivo types.c.

Hace referencia a lnode_::code, ERROR_NONE, y MAX_VECTOR.

Referenciado por debug_start(), y main().

4.34.3.6. int len (linkedlist *l*)

Obtiene la longitud de una lista

Parámetros:

l Lista enlazada

Devuelve:

La longitud

Definición en la línea 57 del archivo types.c.

Hace referencia a lnode_::code.

Referenciado por debug_show_log().

4.34.3.7. void printlist (linkedlist *l*)

Imprime una lista

Parámetros:

l Lista a imprimir

Definición en la línea 20 del archivo types.c.

Hace referencia a lnode_::code.

Referenciado por main().

4.34.3.8. int setcode (linkedlist *l*, int *index*, int *c*)

Fija un codigo en un nodo

Parámetros:

l Lista enlazada

index Indice del nodo

c Codigo deseado

Definición en la línea 45 del archivo types.c.

Hace referencia a lnode_::code, y ERROR_NONE.

Referenciado por debug_start(), y main().

4.34.3.9. int setmsg (linkedlist *l*, int *index*, char *m*[TEXT_LENGTH])

Fija un mensaje en un nodo

Parámetros:

l Lista enlazada

index Indice del nodo

m Mensaje deseado

Definición en la línea 51 del archivo types.c.

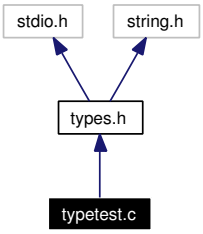
Hace referencia a ERROR_NONE.

Referenciado por debug_start(), y main().

4.35. Referencia del Archivo typetest.c

```
#include "types.h"
```

Dependencia gráfica adjunta para typetest.c:



Funciones

- int main ()

4.35.1. Documentación de las funciones

4.35.1.1. int main ()

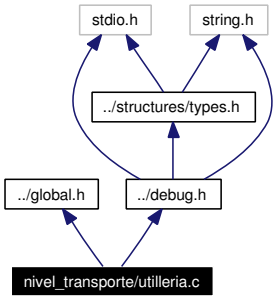
Definición en la línea 3 del archivo typetest.c.

Hace referencia a addnode(), initlist(), printlist(), setcode(), y setmsg().

4.36. Referencia del Archivo utilleria.c

```
#include "../global.h"
#include "../debug.h"
```

Dependencia gráfica adjunta para nivel_transporte/utilleria.c:



Funciones

- void reverse (char s[ ])
- void itoa (int n, char s[ ])
- long summarize (char *data)
- int summarize_chunk (unsigned char *partdata)

4.36.1. Documentación de las funciones

4.36.1.1. void itoa (int *n*, char *s*[ ])

Convierte un entero en una cadena de caracteres

Parámetros:

n El entero

s Cadena a rellenar

Definición en la línea 17 del archivo nivel_transporte/utilleria.c.

Hace referencia a reverse().

Referenciado por main(), y presenta_socket_enviar().

4.36.1.2. void reverse (char *s*[ ])

Invierte una cadena

Parámetros:

s La cadena

Definición en la línea 5 del archivo nivel_transporte/utilleria.c.

Referenciado por itoa(), y ltoa().

4.36.1.3. long summarize (char * *data*)

Devuelve un numero resumen de una cadena de texto

Parámetros:
data La cadena a resumir

Devuelve:
Numero resumen

Definición en la línea 38 del archivo nivel_transporte/utilleria.c.

Referenciado por enviar_udp(), hash_conv(), y receive_parted().

4.36.1.4. int summarize_chunk (unsigned char * *partdata*)

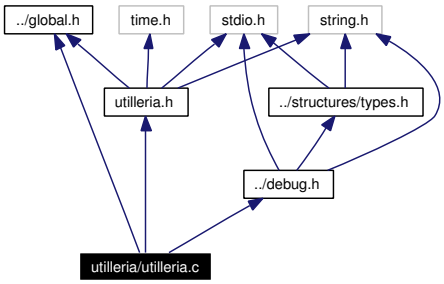
Definición en la línea 49 del archivo nivel_transporte/utilleria.c.

Hace referencia a CHUNK_SIZE.

4.37. Referencia del Archivo utilleria.c

```
#include "utilleria.h"
#include "../debug.h"
#include "../global.h"
```

Dependencia gráfica adjunta para utilleria/utilleria.c:



Funciones

- void reverse (char s[ ])
- void ltoa (long n, char s[ ])
- FILE * fichero_log ()

4.37.1. Documentación de las funciones

4.37.1.1. FILE* fichero_log ()

Devuelve un puntero al archivo de log

Funcion que devuelve un puntero a un archivo que sera el archivo de Log. Sino existe lo crea la primera vez que se llama la funcion y a partir de ahi siempre usa el mismo

Devuelve:
El puntero al fichero de log

Definición en la línea 75 del archivo utilleria/utilleria.c.

4.37.1.2. void ltoa (long *n*, char *s*[ ])

Convierte un long en una cadena de caracteres

Parámetros:
n El long

s La cadena a rellenar

Definición en la línea 46 del archivo utilleria/utilleria.c.

Hace referencia a reverse().

Referenciado por enviar_udp().

4.37.1.3. void reverse (char *s*[ ])

Invierte una cadena

Parámetros:

s La cadena

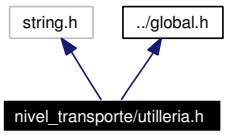
Definición en la línea 5 del archivo nivel_transporte/utilleria.c.

Referenciado por itoa(), y ltoa().

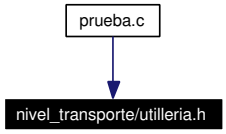
4.38. Referencia del Archivo utilleria.h

```
#include <string.h>
#include "../global.h"
```

Dependencia gráfica adjunta para nivel_transporte/utilleria.h:



Este gráfico muestra que archivos directa o indirectamente incluyen a este archivo:



Funciones

- void reverse (char s[ ])
- void itoa (int n, char s[ ])
- long summarize (char *data)
- int summarize_chunk (char *partdata)

4.38.1. Documentación de las funciones

4.38.1.1. void itoa (int *n*, char *s*[ ])

Convierte un entero en una cadena de caracteres

Parámetros:

n El entero

s Cadena a rellenar

Definición en la línea 17 del archivo nivel_transporte/utilleria.c.

4.38.1.2. void reverse (char *s*[ ])

Invierte una cadena

Parámetros:

s La cadena

Definición en la línea 5 del archivo nivel_transporte/utilleria.c.

4.38.1.3. long summarize (char * *data*)

Devuelve un numero resumen de una cadena de texto

Parámetros:
data La cadena a resumir

Devuelve:
Numero resumen

Definición en la línea 38 del archivo nivel_transporte/utilleria.c.

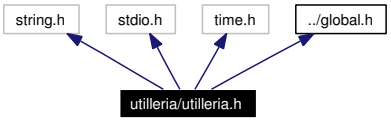
4.38.1.4. int summarize_chunk (char * *partdata*)

Referenciado por receive_parted(), y send_parted().

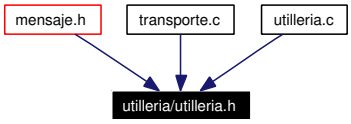
4.39. Referencia del Archivo utilleria.h

```
#include <string.h>
#include <stdio.h>
#include <time.h>
#include "../global.h"
```

Dependencia gráfica adjunta para utilleria/utilleria.h:



Este gráfico muestra que archivos directa o indirectamente incluyen a este archivo:



Funciones

- void itoa (int n, char s[ ])
- void ltoa (long n, char s[ ])
- FILE * fichero_log ()

4.39.1. Documentación de las funciones

4.39.1.1. FILE* fichero_log ()

Devuelve un puntero al archivo de log

Funcion que devuelve un puntero a un archivo que sera el archivo de Log. Sino existe lo crea la primera vez que se llama la funcion y a partir de ahi siempre usa el mismo

Devuelve:
El puntero al fichero de log

Definición en la línea 75 del archivo utilleria/utilleria.c.

4.39.1.2. void itoa (int *n*, char *s*[ ])

Convierte un entero en una cadena de caracteres

Parámetros:

n El entero

s Cadena a rellenar

Definición en la línea 17 del archivo nivel_transporte/utilleria.c.

Hace referencia a reverse().

Referenciado por main(), y presenta_socket_enviar().

4.39.1.3. void ltoa (long *n*, char *s*[ ])

Convierte un long en una cadena de caracteres

Parámetros:

n El long

s La cadena a rellenar

Definición en la línea 46 del archivo utilleria/utilleria.c.

Hace referencia a reverse().

Referenciado por enviar_udp().

Índice alfabético