Client Asincrono

1.3.5

Generato da Doxygen 1.8.12

Indice

Capitolo 1

Pagina Principale

Il programma 'Client_asincrono' tramite un Socket creato nel main, con la funzione 'WSAStartup' e 'socket', poi abbiamo assegnato un porta al canale della Socket tramite la funzione 'connect'. Da li' si avra' una connessione bidirezionale asincrona tra Server e Client, terminabile tramite una sequenza di escape. In caso di errore nella creazione o connesione del Socket il programma terminera' rilasciando un determinato codice d'errore. Inoltre nel progetto utlizzeremo una libreria non citata esplicitamente nel codice, ma individuabile nelle propieta'del progetto, la libreria "Ws2_32.lib".

Capitolo 2

Indice delle strutture dati

2.1 Strutture dati

Queste sono le strutture dati con una loro breve descrizione:

Capitolo 3

Indice dei file

3.1 Elenco dei file

Questo è un elenco di tutti i file con una loro breve descrizione:

Capitolo 4

Documentazione delle classi

4.1 Riferimenti per la struct Str

Campi

- SOCKET Client_socket
- char msg_c_recv [MAX_BUFFER]
- char msg_c_send [MAX_BUFFER]

4.1.1 Descrizione dettagliata

La struct Str contiene tutte le variabili utili all'utilizzo della cominicazione

4.1.2 Documentazione dei campi

4.1.2.1 Client_socket

```
SOCKET Str::Client_socket
```

Il canale logico di comunicazione

4.1.2.2 msg_c_recv

```
char Str::msg_c_recv[MAX_BUFFER]
```

La variabile dedicata alla ricezione del messaggio

4.1.2.3 msg_c_send

```
char Str::msg_c_send[MAX_BUFFER]
```

La variabile dedicata all'invio del messaggio

La documentazione per questa struct è stata generata a partire dal seguente file:

- Origine_client_as.cpp

Capitolo 5

Documentazione dei file

5.1 Riferimenti per il file Origine_client_as.cpp

```
#include <winsock.h>
#include <process.h>
#include <string.h>
#include <iostream>
```

Strutture dati

- struct Str

Definizioni

- #define MAX_BUFFER 100

Funzioni

- unsigned __stdcall Recv (void *param)

  riceve il messaggio dal Server e lo mostra in output.
- unsigned __stdcall Send (void *param)

  Invia il messaggio al Server.
- int __cdecl main (void)

  Inizializziamo il Socket e li diamo un MAKEWORD(2.0, 2.0) per indicare la versione di Windows che il Socket andra' ad utilizzare, tutt'ora la versione piu' recente e' la "2.2". Il nostro Socket di "connesione" 'clientsocket' e' un IPv4, con connessione sicura, e protocollo TCP. Il main dopo aver fatto partire i relativi Thread si blocca alla linea '97' tramite un 'WaitForMultipleObject' dove attende la fine o da parte del Server o del Client tramite una sequenza di escape uguale alla lettera 'q'. In caso di errore nella creazione o connesione del Socket il programma terminera' non prima di lasciare, in output, un codice relativo all'errore tramite la funzione 'WSAGetLastError()'.

5.1.1 Documentazione delle definizioni

5.1.1.1 MAX_BUFFER

```
#define MAX_BUFFER 100
```

La struct Str contiene tutte le variabili utili all'utilizzo della cominicazione

5.1.2 Documentazione delle funzioni

5.1.2.1 main()

```
int __cdecl main (
              void  )
```

Inizializziamo il Socket e li diamo un MAKEWORD(2.0, 2.0) per indicare la versione di Windows che il Socket andra' ad utilizzare, tutt'ora la versione piu' recente e' la "2.2". Il nostro Socket di "connesione" 'clientsocket' e' un IPv4, con connessione sicura, e protocollo TCP. Il main dopo aver fatto partire i relativi Thread si blocca alla linea '97' tramite un 'WaitForMultipleObject' dove attende la fine o da parte del Server o del Client tramite una sequenza di escape uguale alla lettera 'q'. In caso di errore nella creazione o connesione del Socket il programma terminera' non prima di lasciare, in output, un codice relativo all'errore tramite la funzione 'WSAGetLastError()'.

```
...
TH[0] = (HANDLE)_beginthreadex(NULL, 0, &Recv, &S, 0, 0);
TH[1] = (HANDLE)_beginthreadex(NULL, 0, &Send, &S, 0, 0);

printf("Connessione con il Server...\n");
printf("Client\t \t \tServer\n");

WaitForMultipleObjects(2, TH, TRUE, INFINITE);
...
```

Autore

tarchi.giacomo

5.1.2.2 Recv()

```
unsigned __stdcall Recv (
              void * param )
```

riceve il messaggio dal Server e lo mostra in output.

Parametri

*void**	data: Contiene la struttura Str

Restituisce

unsigned: zero.

```
{
```

```
    Str* S = (Str*)param;

    while (1)
    {
        if (recv(S->Client_socket, S->msg_c_recv, sizeof(S->
        msg_c_recv), 0) > 0)
        {
            printf("\t \t \t%s\n", S->msg_c_recv);
            if (strcmp(S->msg_c_recv, "q") == 0 || strcmp(S->msg_c_send, "q") == 0) break
        ; //sequenza di escape
        }
    }
    _endthreadex(0);
    return 0;
}
```

Autore

tarchi.giacomo

5.1.2.3 Send()

```
unsigned __stdcall Send (
            void * param )
```

Invia il messaggio al Server.

Parametri

*void**	data: Contiene la struttura Str

Restituisce

unsigned: zero.

```
{
    Str* S = (Str*)param;

    while (1)
    {
        std::cin.getline(S->msg_c_send, MAX_BUFFER);
        send(S->Client_socket, S->msg_c_send, sizeof(S->
        msg_c_send), 0);
        if (strcmp(S->msg_c_recv, "q") == 0 || strcmp(S->msg_c_send, "q") == 0) break; //
        sequenza di escape
    }
    _endthreadex(0);
    return 0;
}
```

Autore

tarchi.giacomo

Indice analitico