MyFTP

- Camilla Cacace (camilla.cacace001@studenti.uniparthenope.it)
- Ciro Angarella (ciro.angarella001@studenti.uniparthenope.it)
- Vincenzo Terracciano (vincenzo.terracciano003@studenti.uniparthenope.it)

Generated by Doxygen 1.9.1

Chapter 1

Data Structure Index

1.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

Chapter 3

Data Structure Documentation

3.1 USER Struct Reference

Rappresenta la struttura di un utente nel programma.

Data Fields

- char name [20]

  Il nome dell'utente.
- char pass [20]

  La password del profilo.
- int log_state

  Valore booleano che rappresenta lo stato di login dell'utente.
- int clientSocket

  fd del processo associato all'utente.
- char * directoryPath

  Stringa che contiene il percorso della directory dell'utente.
- char rename_from [BUFFER_SIZE]

  variabile usata per salvare il path del file che si vuole rinominare

3.1.1 Detailed Description

Rappresenta la struttura di un utente nel programma.

Quando un nuovo client si connette al server, il suo fd non viene assocciato a nessun user ed è considerato quindi come un user anonimo.Quando immette il comando "user" il sup fd viene associato al nome inserito e quando effettua l'accesso con il comando "pass" lo stato dell'user diventa loggato. Al momento dell'uso del comando "quit" il suo fd e il suo log_state vengono riportati ai valori di default: 0 per log_state e -1 per clientSocket.

3.1.2 Field Documentation

3.1.2.1 clientSocket

```
int USER::clientSocket
```

fd del processo associato all'utente.

3.1.2.2 directoryPath

```
char* USER::directoryPath
```

Stringa che contiene il percorso della directory dell'utente.

3.1.2.3 log_state

```
int USER::log_state
```

Valore booleano che rappresenta lo stato di login dell'utente.

3.1.2.4 name

```
char USER::name[20]
```

Il nome dell'utente.

Può contenere venti caratteri ed è il parametro cercato con il comando "user".

3.1.2.5 pass

```
char USER::pass[20]
```

La password del profilo.

Può contenere venti caratteri ed è il parametro cercato con il comando "pass".

3.1.2.6 rename_from

```
char USER::rename_from[BUFFER_SIZE]
```

variabile usata per salvare il path del file che si vuole rinominare

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

- /home/angalinux/myFTP/server.c

Chapter 4

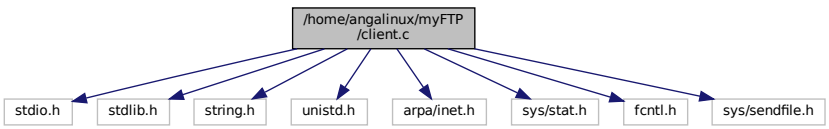
File Documentation

4.1 /home/angalinux/myFTP/client.c File Reference

dettagli implementativi clienntt

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <arpa/inet.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <sys/sendfile.h>
```

Include dependency graph for client.c:



Macros

- #define PORT 50000

  porta alla quale si collega il servizio
- #define BUFFER_SIZE 1024

  dimensione del buffer usato dai buffer del client

Functions

- int setupAndConnectDataSocket (char ∗port_responsed)

  Configura e stabilisce la connessione per il socket dati.
- int main ()

  Funzione principale del programma client.

4.1.1 Detailed Description

dettagli implementativi cliennt

Author

Camilla Cacace (camilla.cacace001@studenti.uniparthenope.it)
Ciro Angarella (ciro.angarella001@studenti.uniparthenope.it)
Vincenzo Terracciano (vincenzo.terracciano003@studenti.uniparthenope.it)

il client dovendo comunicare con un unico server, viene implementato con un sistema di I/O bloccante, ad ogni richiesta inviata al server, il client ne aspetta la risposta sospendendo quindi la possibilità di mandare una seconda richiesta al server. Dopo aver mandato il comando al server (command_buffer), questa stringa viene divisa nella command_word e nella stringa arg. La command word viene utilizzata per sapere in quale state del PI entrare, eseguendo successivamente la connessione sul DTP del server, e ricevere succesivamente l'output del server.

Version

0.1

Date

2024-02-01

Copyright

4.1.2 Macro Definition Documentation

4.1.2.1 BUFFER_SIZE

```
#define BUFFER_SIZE 1024
```

dimensione del buffer usato dai buffer del client

4.1.2.2 PORT

```
#define PORT 50000
```

porta alla quale si collega il servizio

4.1.3 Function Documentation

4.1.3.1 main()

```
int main ( )
```

Funzione principale del programma client.

La funzione crea un socket, si connette a un server remoto, e poi avvia un ciclo di comunicazione interattiva con il server. Per ogni iterazione del ciclo, il client invia una stringa al server e riceve la porta sulla quale deve collegarsi per il data transfer. Successivamente, crea un data socket e si connette al server su tale porta per ricevere dati.

4.1.3.2 setupAndConnectDataSocket()

```
int setupAndConnectDataSocket (
            char * port_responsed )
```

Configura e stabilisce la connessione per il socket dati.

Questa funzione si occupa di configurare e stabilire la connessione per il socket dati. Prende come parametro la porta ricevuta come risposta dal server.

Parameters

port_responsed	La porta ricevuta come risposta dal server.

Returns

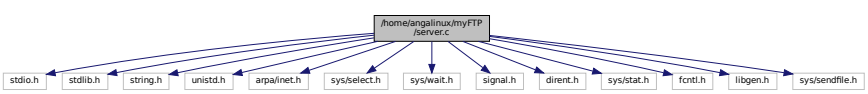
Il descrittore del socket dati.

4.2 /home/angalinux/myFTP/server.c File Reference

dettagli implementativi server

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <arpa/inet.h>
#include <sys/select.h>
#include <sys/wait.h>
#include <signal.h>
#include <dirent.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <libgen.h>
#include <sys/sendfile.h>
```

Include dependency graph for server.c:



Data Structures

- struct USER

  Rappresenta la struttura di un utente nel programma.

Macros

- #define BUFFER_SIZE 1024

  Dimensione massima usata dai buffer del server.
- #define PORT 50000

  Porta dove vengono connessi i client e dove arrivvano i comandi dell'utente.
- #define DATA_PORT 50001

  Porta dove vengono trasferiti gli output e usata per la trasmissione dei file tra server e client.
- #define BACKLOG 4
- #define MAX_USER 3

  Dimensione della struttura hardcoded che contiene gli user registrati.

Functions

- void serverPI (char ∗command, int dataSocket, int clientSocket, fd_set command_fds, int fd_clients_↩ sockets[ ], int i)

  Gestisce i comandi inviati dal client attraverso la connessione di controllo.
- int ricercaPerNome (struct USER array[ ], int lunghezza, char ∗arg)

  Ricerca un utente in un array per nome.
- int ricercaPerFd (struct USER array[ ], int lunghezza, int fd)

  Ricerca un utente in un array per descrittore di file (fd).
- void sendFileList (int dataSocket, char ∗directoryPath)

  Contiene la definizione della funzione per inviare la lista dei file.
- int main ()

Variables

- struct USER registered_user [MAX_USER]

  Questa è la struttura degli user registrati hardcoded all'interno del server.
- char ∗ anonDir = "/home/angalinux/Desktop/FTPpath/anon"

  directory per tutti gli utenti anonimi

4.2.1 Detailed Description

dettagli implementativi server

Author

Camilla Cacace (camilla.cacace001@studenti.uniparthenope.it)
Ciro Angarella (ciro.angarella001@studenti.uniparthenope.it)
Vincenzo Terracciano (vincenzo.terracciano003@studenti.uniparthenope.it)

il server implementato gestisce le richieste degli utenti utilizzando il multiplex attraverso la funzione select(), così da poter monitorare contemporaneamnete più descrittori di diversi client. L'implemetazione del server FTP avviene attraverso l'uso di due porte, "PORT" dove arrivano le richeiste al servizio dei client e la porta "DATA_PORT" per il trasferimento dei file e dei dati di output. Dopo che il multiplex verifica il fd del client da servire, legge il la stringa inviata dal client e la inserisce nella funzione DPI che ne interpetrerà i comandi. Il DPI intepetrato il comando, eseguirà le funzioni del DTP necessarie per stabilere una connesione sulla DATA_PORT e inviare il relativo output.

Version

0.1

Date

2024-02-01

Copyright

4.2.2 Macro Definition Documentation

4.2.2.1 BACKLOG

```
#define BACKLOG 4
```

4.2.2.2 BUFFER_SIZE

```
#define BUFFER_SIZE 1024
```

Dimensione massima usata dai buffer del server.

4.2.2.3 DATA_PORT

```
#define DATA_PORT 50001
```

Porta dove vengono trasferiti gli output e usata per la trasmissione dei file tra server e client.

4.2.2.4 MAX_USER

```
#define MAX_USER 3
```

Dimensione della struttura hardcoded che contiene gli user registrati.

4.2.2.5 PORT

```
#define PORT 50000
```

Porta dove vengono connessi i client e dove arrivvano i comandi dell'utente.

4.2.3 Function Documentation

4.2.3.1 main()

```
int main ( )
```

4.2.3.2 ricercaPerFd()

```
int ricercaPerFd (
            struct USER array[],
            int lunghezza,
            int fd )
```

Ricerca un utente in un array per descrittore di file (fd).

Questa funzione cerca un utente all'interno di un array di utenti in base al descrittore di file.

Parameters

array	L'array di utenti in cui effettuare la ricerca.
lunghezza	La lunghezza dell'array di utenti.
fd	Il descrittore di file da cercare.

Returns

Restituisce l'indice dell'utente se trovato, altrimenti restituisce -1.

4.2.3.3 ricercaPerNome()

```
int ricercaPerNome (
            struct USER array[],
            int lunghezza,
            char * arg )
```

Ricerca un utente in un array per nome.

Questa funzione cerca un utente all'interno di un array di utenti in base al nome tramite una ricerca sequenziale.

Parameters

array	L'array di utenti in cui effettuare la ricerca.
lunghezza	La lunghezza dell'array di utenti.
arg	Il nome da cercare.

Returns

Restituisce l'indice dell'utente se trovato, altrimenti restituisce -1.

4.2.3.4 sendFileList()

```
void sendFileList (
            int dataSocket,
            char * directoryPath )
```

Contiene la definizione della funzione per inviare la lista dei file.

Invia la lista dei file presenti in una directory attraverso un socket.

Questa funzione apre la directory specificata, legge la lista dei file escludendo "." e "..", e invia la lista come una singola stringa attraverso il socket fornito.

Parameters

dataSocket	Il socket per l'invio della lista dei file.
clientSocket	Il socket del client associato.
directoryPath	Il percorso della directory da cui ottenere la lista dei file.

4.2.3.5 serverPI()

```
void serverPI (
            char * command,
            int dataSocket,
```

```
            int clientSocket,
            fd_set command_fds,
            int fd_clients_sockets[],
            int i )
```

Gestisce i comandi inviati dal client attraverso la connessione di controllo.

Questa funzione interpreta e gestisce i comandi inviati dal client attraverso la connessione di controllo. Manda il numero di porta di "DATA_PORT" al client a cui viene erogato il servizio, aspetta che il client si connette, elabora il comando e invia l'output al client.

Parameters

command	La stringa contenente il comando inviato dal client.
dataSocket	Il descrittore di file associato alla connessione di dati (DTP).
clientSocket	Il descrittore di file associato alla connessione di controllo del client.
command_fds	L'insieme di descrittori di file associati ai comandi dei client.
fd_clients_sockets	Array dei descrittori di file dei client.
i	L'indice corrente nell'array dei descrittori di file dei client (usato per il quit).

4.2.4 Variable Documentation

4.2.4.1 anonDir

```
char* anonDir = "/home/angalinux/Desktop/FTPpath/anon"
```

directory per tutti gli utenti anonimi

4.2.4.2 registered_user

```
struct USER registered_user[MAX_USER]
```

Initial value:

```
= {
                {"enzo", "insalata", 0, -1,"/home/angalinux/Desktop/FTPpath/enzo", ""},
                {"ciro", "marika", 0, -1, "/home/angalinux/Desktop/FTPpath/ciro", ""},
                {"camilla", "FTP", 0, -1, "/home/angalinux/Desktop/FTPpath/camilla", ""},
                }
```

Questa è la struttura degli user registrati hardcoded all'interno del server.

Questa struttura contiene gli utenti registrati al servizio. Questo server permette anche la connessione e utilizzo del servizio a utenti non registrati, utilizzando il servizio in modo anonimo non effettuando l'acessso. Questo programma non utlizza un'utente fittizio anonimo.

Index