RelayServer

v0.1

Generated by Doxygen 1.7.5.1

Contents

Chapter 1

Class Index

1.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 2

Class Index

2.1 Class List

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

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

Chapter 4

Class Documentation

4.1 serverSocket Class Reference

```
#include <ServerSocket.h>
```

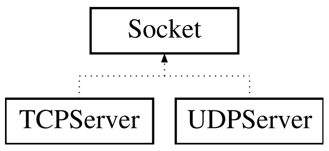
The documentation for this class was generated from the following file:

- **ServerSocket.h**

4.2 Socket Class Reference

```
#include <Socket.h>
```

Inheritance diagram for Socket:



Public Member Functions

- **Socket** ()
- bool **create** (int type)

  Create socket.
- bool **bind** (unsigned short port)

  Bind socket to a port and ipaddress "localhost".
- bool **listen** (int queueLen)

  Bring server up and listen to the specified port for a maximum of connections.

- bool **accept** (**Socket** &)

  Accept new connection, this generates a new socket description.
- bool **disc** (**Socket** &)

  Shutdown socket, and drop all the data received or send.
- bool **connect** (**Socket** &, char ∗, unsigned short port, short int type)

  *Create connection to a **TCPServer** (p. 10).*
- bool **send** (std::string)

  Data Transmission.
- int **recv** (std::string &)
- void **setSocketd** (int dsocket)
- int **getSocketd** ()
- bool **initInter** ()

  Communication interrupts - Setting up the interrupts.
- bool **IsReadable** ()

  Communication interrupts - Read.
- bool **IsWritable** ()

  Communication interrupts - Write.

Private Attributes

- int **socketd**
- fd_set **read_flag**
- fd_set **write_flag**

4.2.1 Constructor & Destructor Documentation

4.2.1.1 Socket::Socket ( )

4.2.2 Member Function Documentation

4.2.2.1 bool Socket::accept (Socket & *new_socket*)

Accept new connection, this generates a new socket description.

4.2.2.2 bool Socket::bind (unsigned short *port*)

Bind socket to a port and ipaddress "localhost".

Parameters

port	bind socket to this port

4.2.2.3 bool Socket::connect (Socket & *dsocket,* char ∗ *ip,* unsigned short *port,* short int *type*)

Create connection to a **TCPServer** (p. 10).

Parameters

char	∗ IP address of the server to connect to
port	Port number of the server to connect to

4.2.2.4 bool Socket::create (int *type*)

Create socket.

Parameters

type	0 for TCP server 1 for UDP server

Note: UDP server not implemented yet.

4.2.2.5 bool Socket::disc (Socket & *new_socket*)

Shutdown socket, and drop all the data received or send.

4.2.2.6 int Socket::getSocketd ( ) `[inline]`

4.2.2.7 bool Socket::initInter ( )

Communication interrupts - Setting up the interrupts.

Note: Further implementation

4.2.2.8 bool Socket::IsReadable ( )

Communication interrupts - Read.

Note: Further implementation, interrupts in read communication.

4.2.2.9 bool Socket::IsWritable ( )

Communication interrupts - Write.

Note: Further implementation, interrupts in the write communication.

4.2.2.10 bool Socket::listen (int *queueLen*)

Bring server up and listen to the specified port for a maximum of connections.

Parameters

queueLen	define the maximum number of connections allowed

4.2.2.11 int Socket::recv (std::string & *s*)

4.2.2.12 bool Socket::send (std::string *s*)

Data Transmission.

4.2.2.13 void Socket::setSocketd (int *dsocket*) `[inline]`

4.2.3 Member Data Documentation

4.2.3.1 fd_set Socket::read_flag `[private]`

4.2.3.2 int Socket::socketd `[private]`

4.2.3.3 fd_set Socket::write_flag `[private]`

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

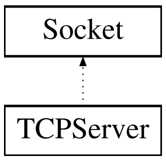
- **Socket.h**
- **Socket.cpp**

4.3 TCPServer Class Reference

TCPServer (p. 10) Class.

`#include <SocketTCP.h>`

Inheritance diagram for TCPServer:



Public Member Functions

- **TCPServer** (unsigned short port, int queue)
  Simple socket object contructor.
- **TCPServer** ()
- **TCPServer** & **operator**>> (std::string)

Operator >> is overwrite to write.

- **TCPServer** & **operator**<< (std::string &)

  Operator << is overwrite to read.
- void **accept** (**TCPServer** &)
- void **disc** (**TCPServer** &)

  Shutdown socket.
- void **connect** (**TCPServer** &, char ∗, unsigned short port)

  Create connection to a ***TCPServer*** *(p. 10).*

4.3.1 Detailed Description

TCPServer (p. 10) Class.

This class initialize and makes the communication between sockets.

4.3.2 Constructor & Destructor Documentation

4.3.2.1 TCPServer::TCPServer (unsigned short *port*, int *queue*)

Simple socket object contructor.

brief **TCPServer** (p. 10) socket constructor

Parameters

port	port which server will be listen.
queue	define queue of allwoing connections to the server.

Description: Initialization of the server. 1 - Create **Socket** (p. 7) 2 - Bind to port and IP address 3 - Listen

4.3.2.2 TCPServer::TCPServer ( ) `[inline]`

4.3.3 Member Function Documentation

4.3.3.1 void TCPServer::accept (**TCPServer** & *new_socket*)

4.3.3.2 void TCPServer::connect (**TCPServer** & *dsocket*, char ∗ *ip*, unsigned short *port*)

Create connection to a **TCPServer** (p. 10).

Parameters

char	∗ IP address of the server to connect to
port	Port number of the server to connect to

4.3.3.3 void TCPServer::disc (**TCPServer** & *new_socket*)

Shutdown socket.

Note: shutdown() used instead of close() since the function shutdown gives more control over what happen to the data in the socket to be closed.

4.3.3.4 **TCPServer** & TCPServer::operator<< (std::string & *s*)

Operator << is overwrite to read.

Parameters

string	string to allocate received data. Accept new connection to the **TCP-Server** (p. 10).

4.3.3.5 **TCPServer** & TCPServer::operator>> (std::string *s*)

Operator >> is overwrite to write.

Parameters

string	string to be send.

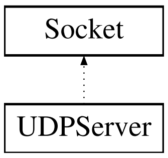
The documentation for this class was generated from the following files:

- **SocketTCP.h**
- **SocketTCP.cpp**

4.4 UDPServer Class Reference

```
#include <SocketUDP.h>
```

Inheritance diagram for UDPServer:



Public Member Functions

- **UDPServer** (unsigned short port)
- **UDPServer** ()

- **UDPServer** & **operator**>> (std::string)
- **UDPServer** & **operator**<< (std::string &)
- void **disc** (**UDPServer** &)
- void **connect** (**UDPServer** &)

4.4.1 Constructor & Destructor Documentation

4.4.1.1 UDPServer::UDPServer (unsigned short *port*)

4.4.1.2 UDPServer::UDPServer ( ) `[inline]`

4.4.2 Member Function Documentation

4.4.2.1 void UDPServer::connect (UDPServer & *dsocket*)

4.4.2.2 void UDPServer::disc (UDPServer & *new_socket*)

4.4.2.3 UDPServer & UDPServer::operator<< (std::string & *s*)

4.4.2.4 UDPServer & UDPServer::operator>> (std::string *s*)

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

- **SocketUDP.h**
- **SocketUDP.cpp**

Chapter 5

File Documentation

5.1 Relay Server.cpp File Reference

`#include "SocketTCP.h" #include "SocketUDP.h" #include <iostream> #include <sys/socket.h> #include <netdb.h>` ×
`#include <string> #include <pthread.h>`

Defines

- #define **BOARD_IP** "127.0.0.1"

  < Localhost is used in this example since daemon is not yet implemented
- #define **BOARD_PORT** 5555

  Port to connect to daemon.

Functions

- void ∗ **tcpTh** (void ∗)
- int **main** ()
- void ∗ **udpTh** (void ∗)

5.1.1 Define Documentation

5.1.1.1 #define BOARD_IP "127.0.0.1"

< Localhost is used in this example since daemon is not yet implemented

EA-LPC2478 Board IP address daemon

5.1.1.2 #define BOARD_PORT 5555

Port to connect to daemon.

Prototyping threads procedures.

5.1.2 Function Documentation

5.1.2.1 int main ( )

5.1.2.2 void ∗ tcpTh (void ∗)

Fork new process in parallel, this creates a new process with a different ID (Important to exit the process when done "exit(0)")

5.1.2.3 void∗ udpTh (void ∗)

5.2 RelayTest.cpp File Reference

5.3 ServerSocket.h File Reference

Classes

- class **serverSocket**

5.4 Socket.cpp File Reference

```
#include "Socket.h" #include <sys/types.h> #include <sys/socket.-
h> #include <sys/time.h> #include <netinet/in.h> #include
<netdb.h> #include <stdio.h> #include <string.h> #include
<unistd.h> #include <stdlib.h> #include <fcntl.h> #include
"string"
```

5.5 Socket.h File Reference

```
#include <sys/types.h>  #include <sys/socket.h>  #include
<netinet/in.h>  #include <netdb.h>  #include <unistd.h> ×
#include <string> #include <arpa/inet.h>
```

Classes

- class **Socket**

Defines

- #define **MAX_BUF** 1000

5.5.1 Define Documentation

5.5.1.1 #define MAX_BUF 1000

5.6 SocketTCP.cpp File Reference

`#include "SocketTCP.h" #include <iostream>`

5.7 SocketTCP.h File Reference

`#include "Socket.h"`

Classes

- class **TCPServer**
  TCPServer *(p. 10) Class.*

5.8 SocketUDP.cpp File Reference

`#include "SocketUDP.h" #include <iostream>`

5.9 SocketUDP.h File Reference

`#include "Socket.h"`

Classes

- class **UDPServer**