

My Project

Generated by Doxygen 1.9.4

1 Multimedia Management System	1
1.1 Description	1
1.2 Features	1
1.3 Prerequisites	1
1.4 Installation	1
1.4.1 1. Installing mpv (Required)	1
1.4.1.1 For Ubuntu/Debian:	2
1.4.1.2 For Fedora:	2
1.4.1.3 For macOS (using Homebrew):	2
1.4.1.4 For Windows:	2
1.5 Building the Application	2
1.5.1 Server Setup	2
1.5.2 Client Setup	2
1.6 Usage	2
1.6.1 Server Mode	2
1.6.2 Local Testing Mode	3
1.7 Class Structure	3
1.8 Serialization	3
1.9 Example	3
1.9.1 Answers to Specific Questions:	3
2 Hierarchical Index	5
2.1 Class Hierarchy	5
3 Class Index	7
3.1 Class List	7
4 File Index	9
4.1 File List	9
5 Class Documentation	11
5.1 Client Class Reference	11
5.1.1 Constructor & Destructor Documentation	11
5.1.1.1 Client()	11
5.1.2 Member Function Documentation	11
5.1.2.1 main()	11
5.1.2.2 send()	12
5.2 Collection Class Reference	12
5.2.1 Detailed Description	13
5.2.2 Constructor & Destructor Documentation	13
5.2.2.1 ~Collection()	13
5.2.3 Member Function Documentation	13
5.2.3.1 disp()	13
5.2.3.2 get_class_name()	14

5.2.3.3	get_name()	14
5.2.3.4	read()	14
5.2.3.5	write()	15
5.2.4	Friends And Related Function Documentation	15
5.2.4.1	MediaManager	15
5.3	Film Class Reference	16
5.3.1	Detailed Description	17
5.3.2	Constructor & Destructor Documentation	17
5.3.2.1	~Film()	17
5.3.3	Member Function Documentation	17
5.3.3.1	deserial_chapters()	17
5.3.3.2	disp()	18
5.3.3.3	get_chapters()	18
5.3.3.4	get_class_name()	18
5.3.3.5	get_nb_chapters()	19
5.3.3.6	operator=()	19
5.3.3.7	read()	19
5.3.3.8	serial_chapters()	20
5.3.3.9	set_chapters()	20
5.3.3.10	write()	21
5.3.4	Friends And Related Function Documentation	21
5.3.4.1	MediaManager	21
5.4	InputBuffer Struct Reference	21
5.4.1	Constructor & Destructor Documentation	21
5.4.1.1	InputBuffer()	22
5.4.1.2	~InputBuffer()	22
5.4.2	Member Data Documentation	22
5.4.2.1	begin	22
5.4.2.2	buffer	22
5.4.2.3	end	22
5.4.2.4	remaining	22
5.5	MediaManager Class Reference	22
5.5.1	Detailed Description	23
5.5.2	Constructor & Destructor Documentation	23
5.5.2.1	MediaManager()	23
5.5.2.2	~MediaManager()	24
5.5.3	Member Function Documentation	24
5.5.3.1	create_collection()	24
5.5.3.2	create_Film()	24
5.5.3.3	create_photo()	25
5.5.3.4	create_video()	25
5.5.3.5	delete_collection()	26

5.5.3.6 delete_media()	26
5.5.3.7 disp_all()	26
5.5.3.8 disp_collection()	27
5.5.3.9 disp_media()	27
5.5.3.10 play_media()	27
5.5.3.11 read()	28
5.5.3.12 write()	28
5.6 MultiMedia Class Reference	28
5.6.1 Detailed Description	29
5.6.2 Constructor & Destructor Documentation	30
5.6.2.1 MultiMedia() [1/2]	30
5.6.2.2 MultiMedia() [2/2]	30
5.6.2.3 ~MultiMedia()	30
5.6.3 Member Function Documentation	30
5.6.3.1 disp()	30
5.6.3.2 get_class_name()	31
5.6.3.3 get_name()	31
5.6.3.4 get_path()	31
5.6.3.5 read()	31
5.6.3.6 run()	32
5.6.3.7 set_name()	32
5.6.3.8 set_path()	32
5.6.3.9 write()	33
5.7 Photo Class Reference	33
5.7.1 Detailed Description	34
5.7.2 Constructor & Destructor Documentation	35
5.7.2.1 ~Photo()	35
5.7.3 Member Function Documentation	35
5.7.3.1 disp()	35
5.7.3.2 get_class_name()	35
5.7.3.3 get_height()	36
5.7.3.4 get_width()	36
5.7.3.5 read()	36
5.7.3.6 run()	36
5.7.3.7 set_height()	37
5.7.3.8 set_width()	37
5.7.3.9 write()	37
5.7.4 Friends And Related Function Documentation	37
5.7.4.1 MediaManager	37
5.8 ServerSocket Class Reference	38
5.8.1 Detailed Description	38
5.8.2 Constructor & Destructor Documentation	38

5.8.2.1 ServerSocket()	38
5.8.2.2 ~ServerSocket()	38
5.8.3 Member Function Documentation	39
5.8.3.1 accept()	39
5.8.3.2 bind()	39
5.8.3.3 close()	39
5.8.3.4 descriptor()	39
5.8.3.5 isClosed()	40
5.8.3.6 setReceiveBufferSize()	40
5.8.3.7 setReuseAddress()	40
5.8.3.8 setSoTimeout()	40
5.8.3.9 setTcpNoDelay()	40
5.9 Socket Class Reference	41
5.9.1 Detailed Description	42
5.9.2 Member Enumeration Documentation	42
5.9.2.1 Errors	42
5.9.3 Constructor & Destructor Documentation	43
5.9.3.1 Socket() [1/2]	43
5.9.3.2 Socket() [2/2]	43
5.9.3.3 ~Socket()	43
5.9.4 Member Function Documentation	43
5.9.4.1 bind() [1/2]	44
5.9.4.2 bind() [2/2]	44
5.9.4.3 cleanup()	44
5.9.4.4 close()	44
5.9.4.5 connect()	44
5.9.4.6 descriptor()	45
5.9.4.7 getReceiveBufferSize()	45
5.9.4.8 getReuseAddress()	45
5.9.4.9 getSendBufferSize()	45
5.9.4.10 getSoLinger()	45
5.9.4.11 getSoTimeout()	45
5.9.4.12 getTcpNoDelay()	46
5.9.4.13 isClosed()	46
5.9.4.14 receive()	46
5.9.4.15 receiveFrom()	46
5.9.4.16 send()	47
5.9.4.17 sendTo()	47
5.9.4.18 setReceiveBufferSize()	47
5.9.4.19 setReuseAddress()	47
5.9.4.20 setSendBufferSize()	48
5.9.4.21 setSoLinger()	48

5.9.4.22 setSoTimeout()	48
5.9.4.23 setTcpNoDelay()	48
5.9.4.24 shutdownInput()	48
5.9.4.25 shutdownOutput()	48
5.9.4.26 startup()	49
5.9.5 Friends And Related Function Documentation	49
5.9.5.1 ServerSocket	49
5.10 SocketBuffer Class Reference	49
5.10.1 Detailed Description	50
5.10.2 Constructor & Destructor Documentation	51
5.10.2.1 SocketBuffer() [1/2]	51
5.10.2.2 SocketBuffer() [2/2]	51
5.10.2.3 ~SocketBuffer()	51
5.10.3 Member Function Documentation	51
5.10.3.1 read()	51
5.10.3.2 readLine()	52
5.10.3.3 readSeparator()	52
5.10.3.4 retrieveLine()	52
5.10.3.5 setReadSeparator()	52
5.10.3.6 setWriteSeparator()	53
5.10.3.7 socket()	53
5.10.3.8 write()	53
5.10.3.9 writeLine()	53
5.10.3.10 writeSeparator()	54
5.10.4 Member Data Documentation	54
5.10.4.1 in_	54
5.10.4.2 insep_	54
5.10.4.3 insize_	54
5.10.4.4 outsep_	54
5.10.4.5 outsize_	54
5.10.4.6 sock_	54
5.11 SocketCnx Class Reference	55
5.11.1 Detailed Description	55
5.11.2 Constructor & Destructor Documentation	55
5.11.2.1 SocketCnx()	56
5.11.2.2 ~SocketCnx()	56
5.11.3 Member Function Documentation	56
5.11.3.1 processRequests()	56
5.11.4 Member Data Documentation	56
5.11.4.1 server_	56
5.11.4.2 sock_	56
5.11.4.3 sockbuf_	56

5.11.4.4 thread_	57
5.12 TCPServer Class Reference	57
5.12.1 Detailed Description	57
5.12.2 Member Typedef Documentation	57
5.12.2.1 Callback	57
5.12.3 Constructor & Destructor Documentation	57
5.12.3.1 TCPServer()	58
5.12.3.2 ~TCPServer()	58
5.12.4 Member Function Documentation	58
5.12.4.1 run()	58
5.12.5 Friends And Related Function Documentation	58
5.12.5.1 SocketCnx	58
5.12.5.2 TCPLock	59
5.13 Video Class Reference	59
5.13.1 Detailed Description	60
5.13.2 Constructor & Destructor Documentation	60
5.13.2.1 ~Video()	60
5.13.3 Member Function Documentation	60
5.13.3.1 disp()	60
5.13.3.2 get_class_name()	61
5.13.3.3 get_duration()	61
5.13.3.4 read()	61
5.13.3.5 run()	62
5.13.3.6 set_duration()	62
5.13.3.7 write()	62
5.13.4 Friends And Related Function Documentation	62
5.13.4.1 Film	62
5.13.4.2 MediaManager	63
5.14 Window Class Reference	63
5.14.1 Detailed Description	64
5.14.2 Constructor & Destructor Documentation	64
5.14.2.1 Window()	64
5.14.3 Member Function Documentation	64
5.14.3.1 main()	64
6 File Documentation	65
6.1 cpp/ccsocket.cpp File Reference	65
6.2 cpp/ccsocket.h File Reference	65
6.2.1 Macro Definition Documentation	66
6.2.1.1 INVALID_SOCKET	67
6.2.1.2 NO_SIGPIPE_	67
6.2.1.3 SOCKADDR	67

6.2.1.4 SOCKADDR_IN	67
6.2.1.5 SOCKDATA	67
6.2.1.6 SOCKET	67
6.2.1.7 SOCKSIZE	67
6.3 ccsocket.h	68
6.4 cpp/client.cpp File Reference	70
6.4.1 Function Documentation	71
6.4.1.1 main()	71
6.5 cpp/Collection.cpp File Reference	71
6.6 cpp/Collection.h File Reference	71
6.6.1 Typedef Documentation	72
6.6.1.1 MediaPtr	73
6.7 Collection.h	73
6.8 cpp/Film.cpp File Reference	74
6.9 cpp/Film.h File Reference	74
6.10 Film.h	75
6.11 cpp/main.cpp File Reference	76
6.11.1 Macro Definition Documentation	76
6.11.1.1 SERVER_v	76
6.11.2 Function Documentation	77
6.11.2.1 main()	77
6.11.3 Variable Documentation	77
6.11.3.1 PORT	77
6.12 cpp/MediaManager.cpp File Reference	77
6.13 cpp/MediaManager.h File Reference	78
6.13.1 Typedef Documentation	80
6.13.1.1 CollectPtr	80
6.13.1.2 Dict_collection	80
6.13.1.3 Dict_Media	80
6.13.1.4 MediaPtr	80
6.13.1.5 SharCollectPtr	81
6.13.1.6 SharFilmPtr	81
6.13.1.7 SharPhotoPtr	81
6.13.1.8 SharVideoPtr	81
6.14 MediaManager.h	81
6.15 cpp/MultiMedia.cpp File Reference	82
6.16 cpp/MultiMedia.h File Reference	82
6.17 MultiMedia.h	83
6.18 cpp/Photo.h File Reference	84
6.19 Photo.h	85
6.20 cpp/server.cpp File Reference	86
6.20.1 Function Documentation	87

6.20.1.1 main()	87
6.20.2 Variable Documentation	87
6.20.2.1 PORT	87
6.21 cpp/tcpserver.cpp File Reference	87
6.22 cpp/tcpserver.h File Reference	88
6.23 tcpserver.h	89
6.24 cpp/Video.h File Reference	89
6.25 Video.h	90
6.26 README.md File Reference	91
6.27 swing/Client.java File Reference	91
6.28 swing/Window.java File Reference	91
Index	93

Chapter 1

Multimedia Management System

1.1 Description

This application is a comprehensive multimedia management system that allows users to organize and interact with various types of media files including videos, photos, and films. The system implements an object-oriented approach with inheritance hierarchies and provides both a local mode for testing and a server mode for client-server interaction.

1.2 Features

- Create and manage different types of media:
 - Photos with dimensions (width, height)
 - Videos with duration
 - Films with chapters and durations
- Organize media into named collections
- Display detailed information about media and collections
- Play media files using the mpv media player
- Serialize and deserialize data for persistence
- TCP server functionality for remote client access

1.3 Prerequisites

- C++ compiler with C++11 support or later
- mpv media player (required for the playback functionality)

1.4 Installation

1.4.1 1. Installing mpv (Required)

The application depends on mpv for media playback. You must install mpv before using this application.

1.4.1.1 For Ubuntu/Debian:

```
sudo apt update
sudo apt install mpv
```

1.4.1.2 For Fedora:

```
sudo dnf install mpv
```

1.4.1.3 For macOS (using Homebrew):

```
brew install mpv
```

1.4.1.4 For Windows:

1. Download the installer from mpv.io
2. Install mpv and ensure it's added to your system PATH

1.5 Building the Application

1.5.1 Server Setup

1. Go to the repository `cpp` and type `make run` in the terminal to start the server on host port 3331.
2. You can change the port of the server, but make sure to also update it for the client.

1.5.2 Client Setup

1. Go to the repository `swing` and type `make run` in the terminal to start the client (application).

1.6 Usage

1.6.1 Server Mode

The application by default runs in server mode (with the `SERVER_v` define active in `main.cpp`).

1. The server will start on port 3331 and accept the following commands from clients:
 - `FIND_MEDIA [name]` - Display information about a specific media item
 - `FIND_GROUPE [name]` - Display information about a specific collection
 - `PLAY_MEDIA [name]` - Play a specific media item using mpv
 - `DELETE_MEDIA [name]` - Delete a specific media item
 - `DELETE_COLLECTION [name]` - Delete a specific collection
 - `DISP_ALL` - Display information about all media items and collections
2. For commands that end with `[name]`, you should:
 - Click the "CLEAR" button
 - Type the name of the media or collection you want to play or find

1.6.2 Local Testing Mode

To use the local testing mode (requires changing the `SERVER_v` define in `main.cpp`):

1. Comment out the `#define SERVER_v` line in `main.cpp`
2. Type `make run`

This mode demonstrates the basic functionality by creating sample media objects, displaying their information, serializing them to a file, deleting them, and then restoring them by deserialization.

1.7 Class Structure

- **MultiMedia**: Abstract base class for all media types
- **Video**: Class for video files with duration
- **Photo**: Class for image files with dimensions
- **Film**: Enhanced video class with chapter information
- **Collection**: Container class that holds multiple media objects
- **MediaManager**: Manages the creation and operations on media objects and collections
- **TCPServer**: Handles client connections and request processing

1.8 Serialization

The application supports saving and loading media objects and collections from files. The serialization format is text-based for easy debugging.

1.9 Example

The server mode initializes with some sample data:

- A video named "video1"
- A photo named "photo1"
- A film named "Film2"
- A collection named "HAPPY" containing the above items

1.9.1 Answers to Specific Questions:

Step 5 (Polymorphism):

The array must contain pointers (`Multimedia*` or `shared_ptr<Multimedia>`) to avoid slicing and enable polymorphism.

Step 6 (Arrays):

The modifier must copy the data from the external array to ensure encapsulation.

Step 8 (Groups):

Use pointers to avoid duplication and allow sharing between groups (unlike in Java, where references are sufficient).

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

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

Client	11
InputBuffer	21
JFrame	
Window	63
list	
Collection	12
MediaManager	22
MultiMedia	28
Photo	33
Video	59
Film	16
ServerSocket	38
Socket	41
SocketBuffer	49
SocketCnx	55
TCPServer	57

Chapter 3

Class Index

3.1 Class List

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

Client	11
Collection A collection of multimedia objects	12
Film A class representing a film with chapters, derived from Video	16
InputBuffer	21
MediaManager Manages collections of multimedia objects and their operations	22
MultiMedia Represents a multimedia file	28
Photo A class representing a photo file, derived from the MultiMedia base class	33
ServerSocket	38
Socket	41
SocketBuffer	49
SocketCnx Connection with a given client. Each SocketCnx uses a different thread	55
TCPServer	57
Video A class representing a video file, derived from the MultiMedia base class	59
Window	63

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

cpp/ccsocket.cpp	65
cpp/ccsocket.h	65
cpp/client.cpp	70
cpp/Collection.cpp	71
cpp/Collection.h	71
cpp/Film.cpp	74
cpp/Film.h	74
cpp/main.cpp	76
cpp/MediaManager.cpp	77
cpp/MediaManager.h	78
cpp/MultiMedia.cpp	82
cpp/MultiMedia.h	82
cpp/Photo.h	84
cpp/server.cpp	86
cpp/tcpserver.cpp	87
cpp/tcpserver.h	88
cpp/Video.h	89
swing/Client.java	91
swing/Window.java	91

Chapter 5

Class Documentation

5.1 Client Class Reference

Public Member Functions

- [Client](#) (String host, int port) throws UnknownHostException, IOException
- String [send](#) (String request)

Static Public Member Functions

- static void [main](#) (String argv[])

5.1.1 Constructor & Destructor Documentation

5.1.1.1 Client()

```
Client.Client (
    String host,
    int port ) throws UnknownHostException, IOException [inline]
```

Initialise la connexion. Renvoie une exception en cas d'erreur.

5.1.2 Member Function Documentation

5.1.2.1 main()

```
static void Client.main (
    String argv[] ) [inline], [static]
```

Lit une requete depuis le Terminal, envoie cette requete au serveur, recupere sa reponse et l'affiche sur le Terminal.
Noter que le programme bloque si le serveur ne repond pas.

5.1.2.2 send()

```
String Client.send (
    String request ) [inline]
```

Envoie une requete au server et retourne sa reponse. Noter que la methode bloque si le serveur ne repond pas.

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

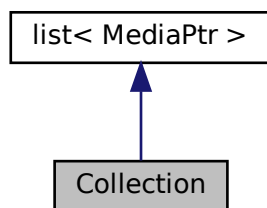
- swing/[Client.java](#)

5.2 Collection Class Reference

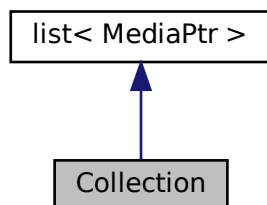
A collection of multimedia objects.

```
#include <Collection.h>
```

Inheritance diagram for Collection:



Collaboration diagram for Collection:



Public Member Functions

- [~Collection](#) ()
Destructor.
- string [get_name](#) () const
Gets the collection name.
- string [get_class_name](#) () const
Gets the class name.
- void [write](#) (ostream &f) const
Serializes the collection to an output stream.
- void [read](#) (istream &f)
Deserializes the collection from an input stream.
- void [disp](#) (ostream &out) const
Displays the collection contents.

Friends

- class [MediaManager](#)

5.2.1 Detailed Description

A collection of multimedia objects.

This class represents a named collection of multimedia items (photos, videos, films) inheriting from `std::list` to provide container functionality. It supports serialization and display of the collection contents.

5.2.2 Constructor & Destructor Documentation

5.2.2.1 ~Collection()

```
Collection::~Collection ( ) [inline]
```

Destructor.

5.2.3 Member Function Documentation

5.2.3.1 disp()

```
void Collection::disp (
    ostream & out ) const
```

Displays the collection contents.

Displays the collection information and its contents.

Parameters

<i>out</i>	Output stream to display to
------------	-----------------------------

Outputs the collection name followed by information about each multimedia object in the collection.

Parameters

<i>out</i>	The output stream to write to
------------	-------------------------------

5.2.3.2 get_class_name()

```
string Collection::get_class_name ( ) const
```

Gets the class name.

Gets the class name identifier.

Returns

"Collection" string

string Always returns "Collection"

5.2.3.3 get_name()

```
string Collection::get_name ( ) const
```

Gets the collection name.

Gets the name of the collection.

Returns

Name of the collection

string The name of the collection

5.2.3.4 read()

```
void Collection::read (
    istream & f )
```

Deserializes the collection from an input stream.

Parameters

<i>f</i>	Input stream to read from
----------	---------------------------

Reads the collection name from the input stream. Assumes the stream is properly formatted (as written by [write\(\)](#)).

Parameters

<i>f</i>	The input stream to read from
----------	-------------------------------

5.2.3.5 write()

```
void Collection::write (  
    ostream & f ) const
```

Serializes the collection to an output stream.

Parameters

<i>f</i>	Output stream to write to
----------	---------------------------

Writes the collection data in a format that can be read back by the [read\(\)](#) method. First writes the collection type marker, then the collection name.

Parameters

<i>f</i>	The output stream to write to
----------	-------------------------------

5.2.4 Friends And Related Function Documentation

5.2.4.1 MediaManager

```
friend class MediaManager [friend]
```

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

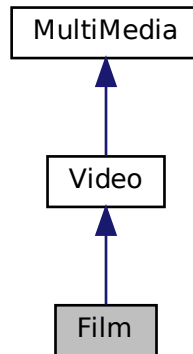
- [cpp/Collection.h](#)
- [cpp/Collection.cpp](#)

5.3 Film Class Reference

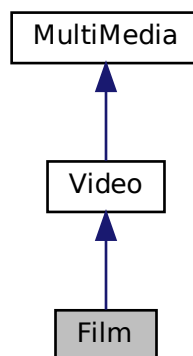
A class representing a film with chapters, derived from [Video](#).

```
#include <Film.h>
```

Inheritance diagram for Film:



Collaboration diagram for Film:



Public Member Functions

- [Film](#) & [operator=](#) (const [Film](#) &from)
Assignment operator.
- [~Film](#) ()

Destructor.

- void [set_chapters](#) (unsigned int const *chapters, unsigned int nb_chapters)

Sets chapter information.

- const unsigned int * [get_chapters](#) () const

Gets chapter durations.

- unsigned int [get_nb_chapters](#) () const

Gets number of chapters.

- void [disp](#) (ostream &out) const override

Displays film information.

- string [get_class_name](#) () const override

Gets class name.

- void [serial_chapters](#) (ostream &f) const

Serializes chapter information.

- void [deserial_chapters](#) (istream &f)

Deserializes chapter information.

- void [write](#) (ostream &f) const override

Serializes film data.

- void [read](#) (istream &f) override

Deserializes film data.

Friends

- class [MediaManager](#)

5.3.1 Detailed Description

A class representing a film with chapters, derived from [Video](#).

This class extends the [Video](#) class to support films with chapters, providing methods to manage chapter information and serialization.

5.3.2 Constructor & Destructor Documentation

5.3.2.1 ~Film()

```
Film::~Film ( )
```

Destructor.

Destructor for [Film](#) class.

Releases memory allocated for chapters array.

5.3.3 Member Function Documentation

5.3.3.1 deserial_chapters()

```
void Film::deserial_chapters (
    istream & f )
```

Deserializes chapter information.

Deserializes chapter durations from input stream.

Parameters

<i>f</i>	Input stream
<i>f</i>	Input stream to read from

5.3.3.2 disp()

```
void Film::disp (
    ostream & out ) const [override], [virtual]
```

Displays film information.

Displays film information including chapters.

Parameters

<i>out</i>	Output stream
<i>out</i>	Output stream to display to

Reimplemented from [MultiMedia](#).

5.3.3.3 get_chapters()

```
const unsigned int * Film::get_chapters ( ) const
```

Gets chapter durations.

Gets the chapter durations array.

Returns

Pointer to chapter durations array

const pointer to chapter durations array

5.3.3.4 get_class_name()

```
string Film::get_class_name ( ) const [override], [virtual]
```

Gets class name.

Gets the class name.

Returns

"Film"

"Film" as string

Implements [MultiMedia](#).

5.3.3.5 get_nb_chapters()

```
unsigned int Film::get_nb_chapters ( ) const
```

Gets number of chapters.

Gets the number of chapters.

Returns

Number of chapters

5.3.3.6 operator=()

```
Film & Film::operator= (
    const Film & from )
```

Assignment operator.

Assignment operator for [Film](#) class.

Parameters

<i>from</i>	Film to assign from
-------------	-------------------------------------

Returns

Reference to the assigned film

Parameters

<i>from</i>	Film object to assign from
-------------	--

Returns

Reference to the assigned [Film](#) object

5.3.3.7 read()

```
void Film::read (
    istream & f ) [override], [virtual]
```

Deserializes film data.

Deserializes film data from input stream.

Parameters

<i>f</i>	Input stream
<i>f</i>	Input stream to read from

Reimplemented from [MultiMedia](#).

5.3.3.8 serial_chapters()

```
void Film::serial_chapters (
    ostream & f ) const
```

Serializes chapter information.

Serializes chapter durations to output stream.

Parameters

<i>f</i>	Output stream
<i>f</i>	Output stream to write to

5.3.3.9 set_chapters()

```
void Film::set_chapters (
    unsigned int const * chapters,
    unsigned int nb_chapters )
```

Sets chapter information.

Sets the chapter durations for the film.

Parameters

<i>chapters</i>	Array of chapter durations
<i>nb_chapters</i>	Number of chapters
<i>chapters</i>	Array of chapter durations
<i>nb_chapters</i>	Number of chapters

Note

Manages memory allocation/deallocation for chapters array

5.3.3.10 write()

```
void Film::write (
    ostream & f ) const [override], [virtual]
```

Serializes film data.

Serializes film data to output stream.

Parameters

<i>f</i>	Output stream
<i>f</i>	Output stream to write to

Reimplemented from [MultiMedia](#).

5.3.4 Friends And Related Function Documentation

5.3.4.1 MediaManager

```
friend class MediaManager [friend]
```

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

- [cpp/Film.h](#)
- [cpp/Film.cpp](#)

5.4 InputBuffer Struct Reference

Public Member Functions

- [InputBuffer](#) (size_t size)
- [~InputBuffer](#) ()

Public Attributes

- char * [buffer](#)
- char * [begin](#)
- char * [end](#)
- [SOCKSIZE](#) remaining

5.4.1 Constructor & Destructor Documentation

5.4.1.1 InputBuffer()

```
InputBuffer::InputBuffer (
    size_t size ) [inline]
```

5.4.1.2 ~InputBuffer()

```
InputBuffer::~~InputBuffer ( ) [inline]
```

5.4.2 Member Data Documentation

5.4.2.1 begin

```
char* InputBuffer::begin
```

5.4.2.2 buffer

```
char* InputBuffer::buffer
```

5.4.2.3 end

```
char* InputBuffer::end
```

5.4.2.4 remaining

```
SOCKSIZE InputBuffer::remaining
```

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

- [cpp/ccsocket.cpp](#)

5.5 MediaManager Class Reference

Manages collections of multimedia objects and their operations.

```
#include <MediaManager.h>
```


Public Member Functions

- [MediaManager](#) ()
Default constructor for [MediaManager](#).
- [~MediaManager](#) ()
Destructor for [MediaManager](#).
- [SharPhotoPtr create_photo](#) (string name, string path, double width, double height)
Creates a new [Photo](#) object.
- [SharVideoPtr create_video](#) (string name, string path, unsigned int duration)
Creates a new [Video](#) object.
- [SharFilmPtr create_Film](#) (string name, string path, unsigned int duration, unsigned int const *chapters, unsigned int nb_chapters)
Creates a new [Film](#) object.
- [SharCollectPtr create_collection](#) (string name)
Creates a new [Collection](#) object.
- void [disp_media](#) (ostream &out, string name) const
Displays information about a specific media object.
- void [disp_collection](#) (ostream &out, string name) const
Displays information about a specific collection.
- void [play_media](#) (string name) const
Plays a media object.
- void [delete_media](#) (string name)
Deletes a media object.
- void [delete_collection](#) (string name)
Deletes a collection.
- void [disp_all](#) (ostream &out) const
Displays information about all media objects and collections.
- void [write](#) (ostream &f) const
Writes the manager's data to an output stream.
- void [read](#) (istream &f)
Reads the manager's data from an input stream.

5.5.1 Detailed Description

Manages collections of multimedia objects and their operations.

This class provides functionality to create, display, play, and delete various types of media objects (photos, videos, films) and collections.

5.5.2 Constructor & Destructor Documentation

5.5.2.1 MediaManager()

```
MediaManager::MediaManager ( ) [inline]
```

Default constructor for [MediaManager](#).

5.5.2.2 ~MediaManager()

```
MediaManager::~MediaManager ( ) [inline]
```

Destructor for [MediaManager](#).

5.5.3 Member Function Documentation

5.5.3.1 create_collection()

```
SharCollectPtr MediaManager::create_collection (
    string name )
```

Creates a new [Collection](#) object.

Creates a new [Collection](#) object and adds it to the groups collection.

Parameters

<i>name</i>	Name of the collection.
-------------	-------------------------

Returns

Shared pointer to the created [Collection](#) object.

5.5.3.2 create_Film()

```
SharFilmPtr MediaManager::create_Film (
    string name,
    string path,
    unsigned int duration,
    unsigned int const * chapters,
    unsigned int nb_chapters )
```

Creates a new [Film](#) object.

Creates a new [Film](#) object and adds it to the media collection.

Parameters

<i>name</i>	Name of the film.
<i>path</i>	File path of the film.
<i>duration</i>	Total duration of the film in seconds.
<i>chapters</i>	Array of chapter durations.
<i>nb_chapters</i>	Number of chapters in the film.

Returns

Shared pointer to the created [Film](#) object.

5.5.3.3 create_photo()

```
SharPhotoPtr MediaManager::create_photo (
    string name,
    string path,
    double width,
    double height )
```

Creates a new [Photo](#) object.

Creates a new [Photo](#) object and adds it to the media collection.

Parameters

<i>name</i>	Name of the photo.
<i>path</i>	File path of the photo.
<i>width</i>	Width of the photo in pixels.
<i>height</i>	Height of the photo in pixels.

Returns

Shared pointer to the created [Photo](#) object.

5.5.3.4 create_video()

```
SharVideoPtr MediaManager::create_video (
    string name,
    string path,
    unsigned int duration )
```

Creates a new [Video](#) object.

Creates a new [Video](#) object and adds it to the media collection.

Parameters

<i>name</i>	Name of the video.
<i>path</i>	File path of the video.
<i>duration</i>	Duration of the video in seconds.

Returns

Shared pointer to the created [Video](#) object.

5.5.3.5 delete_collection()

```
void MediaManager::delete_collection (
    string name )
```

Deletes a collection.

Deletes a collection from the manager.

Parameters

<i>name</i>	Name of the collection to delete.
-------------	-----------------------------------

5.5.3.6 delete_media()

```
void MediaManager::delete_media (
    string name )
```

Deletes a media object.

Deletes a media object from the manager.

Parameters

<i>name</i>	Name of the media object to delete.
-------------	-------------------------------------

Also removes the media from any collections that contain it.

Parameters

<i>name</i>	Name of the media object to delete.
-------------	-------------------------------------

5.5.3.7 disp_all()

```
void MediaManager::disp_all (
    ostream & out ) const
```

Displays information about all media objects and collections.

Displays information about all collections.

Parameters

<i>out</i>	Output stream to display the information.
------------	---

5.5.3.8 disp_collection()

```
void MediaManager::disp_collection (
    ostream & out,
    string name ) const
```

Displays information about a specific collection.

Parameters

<i>out</i>	Output stream to display the information.
<i>name</i>	Name of the collection to display.

5.5.3.9 disp_media()

```
void MediaManager::disp_media (
    ostream & out,
    string name ) const
```

Displays information about a specific media object.

Parameters

<i>out</i>	Output stream to display the information.
<i>name</i>	Name of the media object to display.

5.5.3.10 play_media()

```
void MediaManager::play_media (
    string name ) const
```

Plays a media object.

Plays a media object using its associated player.

Parameters

<i>name</i>	Name of the media object to play.
-------------	-----------------------------------

5.5.3.11 read()

```
void MediaManager::read (
    istream & f )
```

Reads the manager's data from an input stream.

Deserializes media and collections from an input stream.

Parameters

<i>f</i>	Input stream to read the data from.
----------	-------------------------------------

5.5.3.12 write()

```
void MediaManager::write (
    ostream & f ) const
```

Writes the manager's data to an output stream.

Serializes all media and collections to an output stream.

Parameters

<i>f</i>	Output stream to write the data to.
----------	-------------------------------------

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

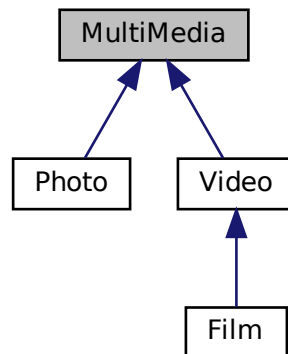
- [cpp/MediaManager.h](#)
- [cpp/MediaManager.cpp](#)

5.6 MultiMedia Class Reference

Represents a multimedia file.

```
#include <MultiMedia.h>
```

Inheritance diagram for MultiMedia:



Public Member Functions

- [MultiMedia](#) ()
Default constructor.
- [MultiMedia](#) (string name, string path)
Constructor with parameters.
- [~MultiMedia](#) ()
Destructor.
- string [get_name](#) () const
Gets the name of the multimedia file.
- string [get_path](#) () const
Gets the file path of the multimedia file.
- void [set_name](#) (string name)
Sets a new name for the multimedia file.
- void [set_path](#) (string path)
Sets a new file path for the multimedia file.
- virtual void [disp](#) (ostream &out) const
Displays multimedia file information.
- virtual void [run](#) () const =0
Runs the file of the multimedia file.
- virtual string [get_class_name](#) () const =0
- virtual void [write](#) (ostream &f) const
Writes the multimedia object's data to an output stream.
- virtual void [read](#) (istream &f)
Reads the multimedia object's data from an input stream.

5.6.1 Detailed Description

Represents a multimedia file.

This class manages a multimedia file by storing its name and file path. It provides methods to modify and display these attributes.

5.6.2 Constructor & Destructor Documentation

5.6.2.1 MultiMedia() [1/2]

```
MultiMedia::MultiMedia ( )
```

Default constructor.

Default constructor for the [MultiMedia](#) class. Initializes a [MultiMedia](#) object with empty name and path.

5.6.2.2 MultiMedia() [2/2]

```
MultiMedia::MultiMedia (
    string name,
    string path )
```

Constructor with parameters.

Parameterized constructor for the [MultiMedia](#) class.

Parameters

<i>name</i>	Name of the multimedia file.
<i>path</i>	File path of the multimedia file.
<i>name</i>	The name of the multimedia object.
<i>path</i>	The file path of the multimedia object.

5.6.2.3 ~MultiMedia()

```
MultiMedia::~MultiMedia ( )
```

Destructor.

Destructor for the [MultiMedia](#) class.

5.6.3 Member Function Documentation

5.6.3.1 disp()

```
void MultiMedia::disp (
    ostream & out ) const [virtual]
```

Displays multimedia file information.

Displays the basic information of the multimedia object.

Parameters

<i>out</i>	Output stream where the information is printed.
------------	---

Outputs the name and path of the multimedia object to the given output stream.

Parameters

<i>out</i>	The output stream where the information will be displayed.
------------	--

Reimplemented in [Film](#), [Photo](#), and [Video](#).

5.6.3.2 get_class_name()

```
virtual string MultiMedia::get_class_name ( ) const [pure virtual]
```

Implemented in [Film](#), [Photo](#), and [Video](#).

5.6.3.3 get_name()

```
string MultiMedia::get_name ( ) const
```

Gets the name of the multimedia file.

Gets the name of the multimedia object.

Returns

The file name as a string.

The name of the multimedia object as a string.

5.6.3.4 get_path()

```
string MultiMedia::get_path ( ) const
```

Gets the file path of the multimedia file.

Gets the path of the multimedia object.

Returns

The file path as a string.

The file path of the multimedia object as a string.

5.6.3.5 read()

```
void MultiMedia::read (
    istream & f ) [virtual]
```

Reads the multimedia object's data from an input stream.

Deserializes the name and path of the multimedia object from the given input stream.

Parameters

<i>f</i>	The input stream from which the data will be read.
----------	--

Reimplemented in [Photo](#), [Video](#), and [Film](#).

5.6.3.6 run()

```
virtual void MultiMedia::run ( ) const [pure virtual]
```

Runs the file of the multimedia file.

Implemented in [Photo](#), and [Video](#).

5.6.3.7 set_name()

```
void MultiMedia::set_name (
    string name )
```

Sets a new name for the multimedia file.

Sets the name of the multimedia object.

Parameters

<i>name</i>	The new file name.
<i>name</i>	The new name to set for the multimedia object.

5.6.3.8 set_path()

```
void MultiMedia::set_path (
    string path )
```

Sets a new file path for the multimedia file.

Sets the path of the multimedia object.

Parameters

<i>path</i>	The new file path.
<i>path</i>	The new file path to set for the multimedia object.

5.6.3.9 write()

```
void MultiMedia::write (
    ostream & f ) const [virtual]
```

Writes the multimedia object's data to an output stream.

Serializes the name and path of the multimedia object to the given output stream.

Parameters

<i>f</i>	The output stream where the data will be written.
----------	---

Reimplemented in [Film](#), [Photo](#), and [Video](#).

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

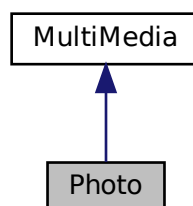
- [cpp/MultiMedia.h](#)
- [cpp/MultiMedia.cpp](#)

5.7 Photo Class Reference

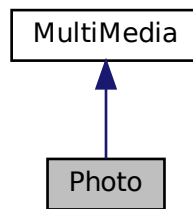
A class representing a photo file, derived from the [MultiMedia](#) base class.

```
#include <Photo.h>
```

Inheritance diagram for Photo:



Collaboration diagram for Photo:



Public Member Functions

- `~Photo ()`
Destructor for the `Photo` class. Outputs a message when the photo object is destroyed.
- void `set_width` (double width)
Sets the width of the photo.
- double `get_width` () const
Gets the width of the photo.
- void `set_height` (double height)
Sets the height of the photo.
- double `get_height` () const
Gets the height of the photo.
- void `disp` (ostream &out) const override
Displays the details of the photo including name, path, width, and height.
- void `run` () const override
Runs the photo using an external player (mpv).
- string `get_class_name` () const override
Gets the class name of the object.
- void `write` (ostream &f) const override
Writes the photo's attributes to an output stream.
- void `read` (istream &f)
Reads the photo's attributes from an input stream.

Friends

- class `MediaManager`

5.7.1 Detailed Description

A class representing a photo file, derived from the `MultiMedia` base class.

This class provides methods to set and get the width and height of the photo, display its details, and run the photo using an external player (mpv).

5.7.2 Constructor & Destructor Documentation

5.7.2.1 ~Photo()

```
Photo::~~Photo ( ) [inline]
```

Destructor for the [Photo](#) class. Outputs a message when the photo object is destroyed.

5.7.3 Member Function Documentation

5.7.3.1 disp()

```
void Photo::disp (
    ostream & out ) const [inline], [override], [virtual]
```

Displays the details of the photo including name, path, width, and height.

This function overrides the [disp\(\)](#) function in the [MultiMedia](#) base class.

Parameters

<i>out</i>	The output stream to which the details are written.
------------	---

Reimplemented from [MultiMedia](#).

5.7.3.2 get_class_name()

```
string Photo::get_class_name ( ) const [inline], [override], [virtual]
```

Gets the class name of the object.

Returns

A string representing the class name ("Photo").

Implements [MultiMedia](#).

5.7.3.3 `get_height()`

```
double Photo::get_height ( ) const [inline]
```

Gets the height of the photo.

Returns

The height of the photo in pixels.

5.7.3.4 `get_width()`

```
double Photo::get_width ( ) const [inline]
```

Gets the width of the photo.

Returns

The width of the photo in pixels.

5.7.3.5 `read()`

```
void Photo::read (
    istream & f ) [inline], [virtual]
```

Reads the photo's attributes from an input stream.

Parameters

<i>f</i>	The input stream to read from.
----------	--------------------------------

Reimplemented from [MultiMedia](#).

5.7.3.6 `run()`

```
void Photo::run ( ) const [inline], [override], [virtual]
```

Runs the photo using an external player (mpv).

This function overrides the [run\(\)](#) function in the [MultiMedia](#) base class. It constructs a system command to run the mpv media player with the photo's path.

Implements [MultiMedia](#).

5.7.3.7 set_height()

```
void Photo::set_height (
    double height ) [inline]
```

Sets the height of the photo.

Parameters

<i>height</i>	The height to set for the photo in pixels.
---------------	--

5.7.3.8 set_width()

```
void Photo::set_width (
    double width ) [inline]
```

Sets the width of the photo.

Parameters

<i>width</i>	The width to set for the photo in pixels.
--------------	---

5.7.3.9 write()

```
void Photo::write (
    ostream & f ) const [inline], [override], [virtual]
```

Writes the photo's attributes to an output stream.

Parameters

<i>f</i>	The output stream to write to.
----------	--------------------------------

Reimplemented from [MultiMedia](#).

5.7.4 Friends And Related Function Documentation

5.7.4.1 MediaManager

```
friend class MediaManager [friend]
```

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

- [cpp/Photo.h](#)

5.8 ServerSocket Class Reference

```
#include <ccsocket.h>
```

Public Member Functions

- [ServerSocket](#) ()
Creates a listening socket that waits for connection requests by TCP/IP clients.
- [~ServerSocket](#) ()
- [Socket](#) * [accept](#) ()
- int [bind](#) (int port, int backlog=50)
- int [close](#) ()
Closes the socket.
- bool [isClosed](#) () const
Returns true if the socket was closed.
- [SOCKET](#) [descriptor](#) ()
Returns the descriptor of the socket.
- int [setReceiveBufferSize](#) (int size)
Sets the SO_RCVBUF option to the specified value.
- int [setReuseAddress](#) (bool)
Enables/disables the SO_REUSEADDR socket option.
- int [setSoTimeout](#) (int timeout)
Enables/disables SO_TIMEOUT with the specified timeout (in milliseconds).
- int [setTcpNoDelay](#) (bool)
Turns on/off TCP coalescence (useful in some cases to avoid delays).

5.8.1 Detailed Description

TCP/IP IPv4 server socket. Waits for requests to come in over the network. TCP/IP sockets do not preserve record boundaries but [SocketBuffer](#) solves this problem.

5.8.2 Constructor & Destructor Documentation

5.8.2.1 ServerSocket()

```
ServerSocket::ServerSocket ( )
```

Creates a listening socket that waits for connection requests by TCP/IP clients.

5.8.2.2 ~ServerSocket()

```
ServerSocket::~~ServerSocket ( )
```


5.8.3 Member Function Documentation

5.8.3.1 accept()

```
Socket * ServerSocket::accept ( )
```

Accepts a new connection request and returns a socket for exchanging data with this client. This function blocks until there is a connection request.

Returns

the new [Socket](#) or nullptr on error.

5.8.3.2 bind()

```
int ServerSocket::bind (
    int port,
    int backlog = 50 )
```

Assigns the server socket to localhost.

Returns

0 on success or a negative value on error, see [Socket::Errors](#)

5.8.3.3 close()

```
int ServerSocket::close ( )
```

Closes the socket.

5.8.3.4 descriptor()

```
SOCKET ServerSocket::descriptor ( ) [inline]
```

Returns the descriptor of the socket.

5.8.3.5 isClosed()

```
bool ServerSocket::isClosed ( ) const [inline]
```

Returns true if the socket was closed.

5.8.3.6 setReceiveBufferSize()

```
int ServerSocket::setReceiveBufferSize (
    int size )
```

Sets the SO_RCVBUF option to the specified value.

5.8.3.7 setReuseAddress()

```
int ServerSocket::setReuseAddress (
    bool state )
```

Enables/disables the SO_REUSEADDR socket option.

5.8.3.8 setSoTimeout()

```
int ServerSocket::setSoTimeout (
    int timeout )
```

Enables/disables SO_TIMEOUT with the specified timeout (in milliseconds).

5.8.3.9 setTcpNoDelay()

```
int ServerSocket::setTcpNoDelay (
    bool state )
```

Turns on/off TCP coalescence (useful in some cases to avoid delays).

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

- [cpp/ccsocket.h](#)
- [cpp/ccsocket.cpp](#)

5.9 Socket Class Reference

```
#include <ccsocket.h>
```

Public Types

- enum `Errors` { `Failed` = -1 , `InvalidSocket` = -2 , `UnknownHost` = -3 }

Public Member Functions

- `Socket` (int type=SOCK_STREAM)
- `Socket` (int type, `SOCKET` sockfd)
Creates a `Socket` from an existing socket file descriptor.
- `~Socket` ()
Destructor (closes the socket).
- int `connect` (const std::string &host, int port)
- int `bind` (int port)
- int `bind` (const std::string &host, int port)
- int `close` ()
Closes the socket.
- bool `isClosed` () const
Returns true if the socket has been closed.
- `SOCKET` `descriptor` ()
Returns the descriptor of the socket.
- void `shutdownInput` ()
Disables further receive operations.
- void `shutdownOutput` ()
Disables further send operations.
- `SOCKSIZE` `send` (const `SOCKDATA` *buf, size_t len, int flags=0)
- `SOCKSIZE` `receive` (`SOCKDATA` *buf, size_t len, int flags=0)
- `SOCKSIZE` `sendTo` (void const *buf, size_t len, int flags, `SOCKADDR` const *to, socklen_t addrlen)
Sends data to a datagram socket.
- `SOCKSIZE` `receiveFrom` (void *buf, size_t len, int flags, `SOCKADDR` *from, socklen_t *addrlen)
Receives data from datagram socket.
- int `setReceiveBufferSize` (int size)
Set the size of the TCP/IP input buffer.
- int `setReuseAddress` (bool)
Enable/disable the SO_REUSEADDR socket option.
- int `setSendBufferSize` (int size)
Set the size of the TCP/IP output buffer.
- int `setSoLinger` (bool, int linger)
Enable/disable SO_LINGER with the specified linger time in seconds.
- int `setSoTimeout` (int timeout)
Enable/disable SO_TIMEOUT with the specified timeout (in milliseconds).
- int `setTcpNoDelay` (bool)
Enable/disable TCP_NODELAY (turns on/off TCP coalescence).
- int `getReceiveBufferSize` () const
Return the size of the TCP/IP input buffer.
- bool `getReuseAddress` () const

- Return SO_REUSEADDR state.*
 - int [getSendBufferSize](#) () const
Return the size of the TCP/IP output buffer.
 - bool [getSoLinger](#) (int &linger) const
Return SO_LINGER state and the specified linger time in seconds.
 - int [getSoTimeout](#) () const
Return SO_TIMEOUT value.
 - bool [getTcpNoDelay](#) () const
Return TCP_NODELAY state.

Static Public Member Functions

- static void [startup](#) ()
- static void [cleanup](#) ()

Friends

- class [ServerSocket](#)

5.9.1 Detailed Description

TCP/IP or UDP/Datagram IPv4 socket. AF_INET connections following the IPv4 Internet protocol are supported.

Note

- [ServerSocket](#) should be used on the server side.
- SIGPIPE signals are ignored when using Linux, BSD or MACOSX.
- TCP/IP sockets do not preserve record boundaries but [SocketBuffer](#) solves this problem.

5.9.2 Member Enumeration Documentation

5.9.2.1 Errors

enum [Socket::Errors](#)

[Socket](#) errors.

- [Socket::Failed](#) (-1): could not connect, could not bind, etc.
- [Socket::InvalidSocket](#) (-2): invalid socket or wrong socket type
- [Socket::UnknownHost](#) (-3): could not reach host

Enumerator

Failed	
InvalidSocket	
UnknownHost	

5.9.3 Constructor & Destructor Documentation

5.9.3.1 Socket() [1/2]

```
Socket::Socket (
    int type = SOCK_STREAM )
```

Creates a new [Socket](#). Creates a AF_INET socket using the IPv4 Internet protocol. Type can be:

- SOCK_STREAM (the default) for TCP/IP connected stream sockets
- SOCK_DGRAM for UDP/datagram sockets (available only on Unix/Linux)

5.9.3.2 Socket() [2/2]

```
Socket::Socket (
    int type,
    SOCKET sockfd )
```

Creates a [Socket](#) from an existing socket file descriptor.

5.9.3.3 ~Socket()

```
Socket::~~Socket ( )
```

Destructor (closes the socket).

5.9.4 Member Function Documentation

5.9.4.1 `bind()` [1/2]

```
int Socket::bind (
    const std::string & host,
    int port )
```

Assigns the socket to an IP address. On Unix/Linux host can be a hostname, on Windows it can only be an IP address.

Returns

0 on success or a negative value on error, see [Socket::Errors](#)

5.9.4.2 `bind()` [2/2]

```
int Socket::bind (
    int port )
```

Assigns the socket to localhost.

Returns

0 on success or a negative value on error, see [Socket::Errors](#)

5.9.4.3 `cleanup()`

```
void Socket::cleanup ( ) [static]
```

5.9.4.4 `close()`

```
int Socket::close ( )
```

Closes the socket.

5.9.4.5 `connect()`

```
int Socket::connect (
    const std::string & host,
    int port )
```

Connects the socket to an address. Typically used for connecting TCP/IP clients to a [ServerSocket](#). On Unix/Linux host can be a hostname, on Windows it can only be an IP address.

Returns

0 on success or a negative value on error which is one of [Socket::Errors](#)

5.9.4.6 descriptor()

```
SOCKET Socket::descriptor ( ) [inline]
```

Returns the descriptor of the socket.

5.9.4.7 getReceiveBufferSize()

```
int Socket::getReceiveBufferSize ( ) const
```

Return the size of the TCP/IP input buffer.

5.9.4.8 getReuseAddress()

```
bool Socket::getReuseAddress ( ) const
```

Return SO_REUSEADDR state.

5.9.4.9 getSendBufferSize()

```
int Socket::getSendBufferSize ( ) const
```

Return the size of the TCP/IP output buffer.

5.9.4.10 getSoLinger()

```
bool Socket::getSoLinger (
    int & linger ) const
```

Return SO_LINGER state and the specified linger time in seconds.

5.9.4.11 getSoTimeout()

```
int Socket::getSoTimeout ( ) const
```

Return SO_TIMEOUT value.

5.9.4.12 getTcpNoDelay()

```
bool Socket::getTcpNoDelay ( ) const
```

Return TCP_NODELAY state.

5.9.4.13 isClosed()

```
bool Socket::isClosed ( ) const [inline]
```

Returns true if the socket has been closed.

5.9.4.14 receive()

```
SOCKSIZE Socket::receive (
    SOCKDATA * buf,
    size_t len,
    int flags = 0 ) [inline]
```

Receives data from a connected (TCP/IP) socket. Reads at most *len* bytes and stores them in *buf*. By default, this function blocks the caller until there is available data.

Returns

the number of bytes that were received, or 0 or [shutdownOutput\(\)](#) was called on the other side, or [Socket::Failed](#) (-1) if an error occurred.

5.9.4.15 receiveFrom()

```
SOCKSIZE Socket::receiveFrom (
    void * buf,
    size_t len,
    int flags,
    SOCKADDR * from,
    socklen_t * addrlen ) [inline]
```

Receives data from datagram socket.

5.9.4.16 send()

```
SOCKSIZE Socket::send (
    const SOCKDATA * buf,
    size_t len,
    int flags = 0 ) [inline]
```

Sends data to a connected (TCP/IP) socket. Sends the first *len* bytes in *buf*.

Returns

the number of bytes that were sent, or 0 or [shutdownInput\(\)](#) was called on the other side, or [Socket::Failed](#) (-1) if an error occurred.

Note

TCP/IP sockets do not preserve record boundaries, see [SocketBuffer](#).

5.9.4.17 sendTo()

```
SOCKSIZE Socket::sendTo (
    void const * buf,
    size_t len,
    int flags,
    SOCKADDR const * to,
    socklen_t addrlen ) [inline]
```

Sends data to a datagram socket.

5.9.4.18 setReceiveBufferSize()

```
int Socket::setReceiveBufferSize (
    int size )
```

Set the size of the TCP/IP input buffer.

5.9.4.19 setReuseAddress()

```
int Socket::setReuseAddress (
    bool state )
```

Enable/disable the SO_REUSEADDR socket option.

5.9.4.20 setSendBufferSize()

```
int Socket::setSendBufferSize (
    int size )
```

Set the size of the TCP/IP output buffer.

5.9.4.21 setSoLinger()

```
int Socket::setSoLinger (
    bool on,
    int linger )
```

Enable/disable SO_LINGER with the specified linger time in seconds.

5.9.4.22 setSoTimeout()

```
int Socket::setSoTimeout (
    int timeout )
```

Enable/disable SO_TIMEOUT with the specified timeout (in milliseconds).

5.9.4.23 setTcpNoDelay()

```
int Socket::setTcpNoDelay (
    bool state )
```

Enable/disable TCP_NODELAY (turns on/off TCP coalescence).

5.9.4.24 shutdownInput()

```
void Socket::shutdownInput ( )
```

Disables further receive operations.

5.9.4.25 shutdownOutput()

```
void Socket::shutdownOutput ( )
```

Disables further send operations.

5.9.4.26 startup()

```
void Socket::startup ( ) [static]
```

initialisation and cleanup of sockets on Widows.

Note

startup is automaticcaly called when a [Socket](#) or a [ServerSocket](#) is created

5.9.5 Friends And Related Function Documentation

5.9.5.1 ServerSocket

```
friend class ServerSocket [friend]
```

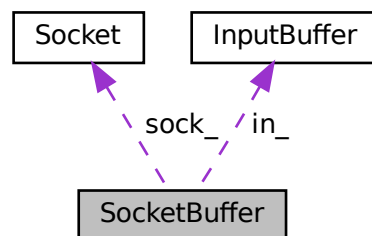
The documentation for this class was generated from the following files:

- [cpp/ccsocket.h](#)
- [cpp/ccsocket.cpp](#)

5.10 SocketBuffer Class Reference

```
#include <ccsocket.h>
```

Collaboration diagram for SocketBuffer:



Public Member Functions

- [~SocketBuffer](#) ()
- [SOCKSIZE readLine](#) (std::string &message)
- [SOCKSIZE writeLine](#) (const std::string &message)
- [SOCKSIZE read](#) (char *buffer, size_t len)
- [SOCKSIZE write](#) (const char *str, size_t len)
- [Socket * socket](#) ()
Returns the associated socket.
- [SocketBuffer](#) (Socket *, size_t inputSize=8192, size_t outputSize=8192)
- [SocketBuffer](#) (Socket &, size_t inputSize=8192, size_t outputSize=8192)
- size_t [insize_](#) {}
- size_t [outsize_](#) {}
- int [insep_](#) {}
- int [outsep_](#) {}
- Socket * [sock_](#) {}
- struct [InputBuffer](#) * [in_](#) {}
- void [setReadSeparator](#) (int separ)
- int [readSeparator](#) () const
- void [setWriteSeparator](#) (int separ)
- int [writeSeparator](#) () const
- bool [retrieveLine](#) (std::string &str, [SOCKSIZE](#) received)

5.10.1 Detailed Description

Preserves record boundaries when exchanging messages between connected TCP/IP sockets. Ensures that one call to [readLine\(\)](#) corresponds to one and exactly one call to [writeLine\(\)](#) on the other side. By default, [writeLine\(\)](#) adds

at the end of each message and [readLine\(\)](#) searches for

, \r or

\r so that it can retrieve the entire record. Beware messages should thus not contain these characters.

```
int main() {
    Socket sock;
    SocketBuffer sockbuf(sock);
    int status = sock.connect("localhost", 3331);
    if (status < 0) {
        cerr << "Could not connect" << endl;
        return 1;
    }
    while (cin) {
        string request, response;
        cout << "Request: ";
        getline(cin, request);
        if (sockbuf.writeLine(request) < 0) {
            cerr << "Could not send message" << endl;
            return 2;
        }
        if (sockbuf.readLine(response) < 0) {
            cerr << "Couldn't receive message" << endl;
            return 3;
        }
    }
    return 0;
}
```

5.10.2 Constructor & Destructor Documentation

5.10.2.1 SocketBuffer() [1/2]

```
SocketBuffer::SocketBuffer (
    Socket * sock,
    size_t inputSize = 8192,
    size_t ouputSize = 8192 )
```

Constructor. *socket* must be a connected TCP/IP [Socket](#). It should **not** be deleted as long as the [SocketBuffer](#) is used. *inputSize* and *ouputSize* are the sizes of the buffers that are used internally for exchanging data.

5.10.2.2 SocketBuffer() [2/2]

```
SocketBuffer::SocketBuffer (
    Socket & sock,
    size_t inputSize = 8192,
    size_t ouputSize = 8192 )
```

5.10.2.3 ~SocketBuffer()

```
SocketBuffer::~SocketBuffer ( )
```

5.10.3 Member Function Documentation

5.10.3.1 read()

```
SOCKSIZE SocketBuffer::read (
    char * buffer,
    size_t len )
```

Reads exactly *len* bytes from the socket, blocks otherwise.

Returns

see [readLine\(\)](#)

5.10.3.2 readLine()

```
SOCKSIZE SocketBuffer::readLine (
    std::string & message )
```

Read a message from a connected socket. `readLine()` receives one (and only one) message sent by `writeLine()` on the other side, ie, a call to `writeLine()` corresponds to one and exactly one call to `readLine()` on the other side. The received data is stored in *message*. This method blocks until the message is fully received.

Returns

The number of bytes that were received or one of the following values:

- 0: shutdownOutput() was called on the other side
- `Socket::Failed` (-1): a connection error occurred
- `Socket::InvalidSocket` (-2): the socket is invalid.

Note

the separator (eg
) is counted in the value returned by `readLine()`.

5.10.3.3 readSeparator()

```
int SocketBuffer::readSeparator ( ) const [inline]
```

5.10.3.4 retrieveLine()

```
bool SocketBuffer::retrieveLine (
    std::string & str,
    SOCKSIZE received ) [protected]
```

5.10.3.5 setReadSeparator()

```
void SocketBuffer::setReadSeparator (
    int separ )
```

Returns/changes the separator used by `readLine()`. `setReadSeparator()` changes the symbol used by `readLine()` to separate successive messages:

- if *separ* < 0 (the default) `readLine()` searches for `\n`, `\r` or `\n\r`.
- if *separ* ≥ 0, `readLine()` searches for this character to separate messages,

5.10.3.6 setWriteSeparator()

```
void SocketBuffer::setWriteSeparator (
    int separ )
```

Returns/changes the separator used by [writeLine\(\)](#). [setWriteSeparator\(\)](#) changes the character(s) used by [writeLine\(\)](#) to separate successive messages:

- if *separ* < 0 (the default) [writeLine\(\)](#) inserts `\n\r` between successive lines.
- if *separ* >= 0, [writeLine\(\)](#) inserts *separ* between successive lines,

5.10.3.7 socket()

```
Socket * SocketBuffer::socket ( ) [inline]
```

Returns the associated socket.

5.10.3.8 write()

```
SOCKSIZE SocketBuffer::write (
    const char * str,
    size_t len )
```

Writes *len* bytes to the socket.

Returns

see [readLine\(\)](#)

5.10.3.9 writeLine()

```
SOCKSIZE SocketBuffer::writeLine (
    const std::string & message )
```

Send a message to a connected socket. [writeLine\(\)](#) sends a message that will be received by a single call of [readLine\(\)](#) on the other side,

Returns

see [readLine\(\)](#)

Note

if *message* contains one or several occurrences of the separator, [readLine\(\)](#) will be called as many times on the other side.

5.10.3.10 writeSeparator()

```
int SocketBuffer::writeSeparator ( ) const [inline]
```

5.10.4 Member Data Documentation

5.10.4.1 in_

```
struct InputBuffer* SocketBuffer::in_ {} [protected]
```

5.10.4.2 insep_

```
int SocketBuffer::insep_ {} [protected]
```

5.10.4.3 insize_

```
size_t SocketBuffer::insize_ {} [protected]
```

5.10.4.4 outsep_

```
int SocketBuffer::outsep_ {} [protected]
```

5.10.4.5 outsize_

```
size_t SocketBuffer::outsize_ {} [protected]
```

5.10.4.6 sock_

```
Socket* SocketBuffer::sock_ {} [protected]
```

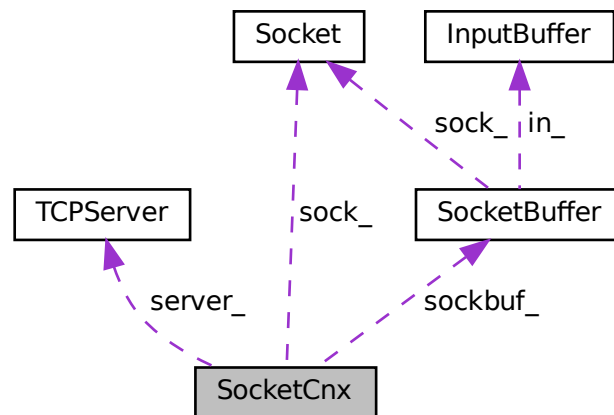
The documentation for this class was generated from the following files:

- [cpp/ccsocket.h](#)
- [cpp/ccsocket.cpp](#)

5.11 SocketCnx Class Reference

Connection with a given client. Each [SocketCnx](#) uses a different thread.

Collaboration diagram for SocketCnx:



Public Member Functions

- [SocketCnx](#) ([TCPServer](#) &, [Socket](#) *)
- [~SocketCnx](#) ()
- void [processRequests](#) ()

Public Attributes

- [TCPServer](#) & `server_`
- [Socket](#) * `sock_`
- [SocketBuffer](#) * `sockbuf_`
- `std::thread` `thread_`

5.11.1 Detailed Description

Connection with a given client. Each [SocketCnx](#) uses a different thread.

5.11.2 Constructor & Destructor Documentation

5.11.2.1 SocketCnx()

```
SocketCnx::SocketCnx (
    TCPServer & server,
    Socket * socket )
```

5.11.2.2 ~SocketCnx()

```
SocketCnx::~~SocketCnx ( )
```

5.11.3 Member Function Documentation

5.11.3.1 processRequests()

```
void SocketCnx::processRequests ( )
```

5.11.4 Member Data Documentation

5.11.4.1 server_

```
TCPServer& SocketCnx::server_
```

5.11.4.2 sock_

```
Socket* SocketCnx::sock_
```

5.11.4.3 sockbuf_

```
SocketBuffer* SocketCnx::sockbuf_
```

5.11.4.4 thread_

```
std::thread SocketCnx::thread_
```

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

- [cpp/tcpserver.cpp](#)

5.12 TCPServer Class Reference

```
#include <tcpserver.h>
```

Public Types

- using [Callback](#) = std::function< bool(std::string const &request, std::string &response) >

Public Member Functions

- [TCPServer](#) ([Callback](#) const &callback)
- virtual [~TCPServer](#) ()
- virtual int [run](#) (int port)

Friends

- class [TCPLock](#)
- class [SocketCnx](#)

5.12.1 Detailed Description

TCP/IP IPv4 server. Supports TCP/IP AF_INET IPv4 connections with multiple clients. One thread is used per client.

5.12.2 Member Typedef Documentation

5.12.2.1 Callback

```
using TCPServer::Callback = std::function< bool(std::string const& request, std::string& response) >
```

5.12.3 Constructor & Destructor Documentation

5.12.3.1 TCPServer()

```
TCPServer::TCPServer (
    Callback const & callback )
```

initializes the server. The callback function will be called each time the server receives a request from a client.

- *request* contains the data sent by the client
- *response* will be sent to the client as a response The connection with the client is closed if the callback returns false.

5.12.3.2 ~TCPServer()

```
TCPServer::~~TCPServer ( ) [virtual]
```

5.12.4 Member Function Documentation

5.12.4.1 run()

```
int TCPServer::run (
    int port ) [virtual]
```

Starts the server. Binds an internal [ServerSocket](#) to *port* then starts an infinite loop that processes connection requests from clients.

Returns

0 on normal termination, or a negative value if the [ServerSocket](#) could not be bound (value is then one of [Socket::Errors](#)).

5.12.5 Friends And Related Function Documentation

5.12.5.1 SocketCnx

```
friend class SocketCnx [friend]
```

5.12.5.2 TCPLock

```
friend class TCPLock [friend]
```

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

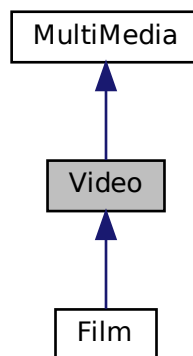
- [cpp/tcpserver.h](#)
- [cpp/tcpserver.cpp](#)

5.13 Video Class Reference

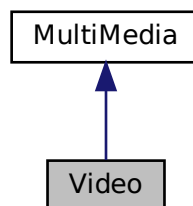
A class representing a video file, derived from the [MultiMedia](#) base class.

```
#include <Video.h>
```

Inheritance diagram for Video:



Collaboration diagram for Video:



Public Member Functions

- [~Video](#) ()
Destructor for the [Video](#) class.
- void [set_duration](#) (unsigned int duration)
Sets the duration of the video.
- unsigned int [get_duration](#) () const
Gets the duration of the video.
- void [disp](#) (ostream &out) const override
Displays the details of the video including name, path, and duration.
- void [run](#) () const override
Plays the video using an external player (mpv).
- string [get_class_name](#) () const override
Gets the class name of the object.
- void [write](#) (ostream &f) const override
Writes the video's attributes to an output stream.
- void [read](#) (istream &f)
Reads the video's attributes from an input stream.

Friends

- class [MediaManager](#)
- class [Film](#)

5.13.1 Detailed Description

A class representing a video file, derived from the [MultiMedia](#) base class.

This class provides methods to set and get the duration of the video, display its details, and run the video using an external player (mpv).

5.13.2 Constructor & Destructor Documentation

5.13.2.1 ~Video()

```
Video::~Video ( ) [inline]
```

Destructor for the [Video](#) class.

5.13.3 Member Function Documentation

5.13.3.1 disp()

```
void Video::disp (
    ostream & out ) const [inline], [override], [virtual]
```

Displays the details of the video including name, path, and duration.

This function overrides the [disp\(\)](#) function in the [MultiMedia](#) base class.

Parameters

<i>out</i>	The output stream to which the details are written.
------------	---

Reimplemented from [MultiMedia](#).

5.13.3.2 get_class_name()

```
string Video::get_class_name ( ) const [inline], [override], [virtual]
```

Gets the class name of the object.

Returns

A string representing the class name ("Video").

Implements [MultiMedia](#).

5.13.3.3 get_duration()

```
unsigned int Video::get_duration ( ) const [inline]
```

Gets the duration of the video.

Returns

The duration of the video in seconds.

5.13.3.4 read()

```
void Video::read (
    istream & f ) [inline], [virtual]
```

Reads the video's attributes from an input stream.

Parameters

<i>f</i>	The input stream to read from.
----------	--------------------------------

Reimplemented from [MultiMedia](#).

5.13.3.5 run()

```
void Video::run ( ) const [inline], [override], [virtual]
```

Plays the video using an external player (mpv).

This function overrides the [run\(\)](#) function in the [MultiMedia](#) base class. It constructs a system command to run the mpv media player with the video's path.

Implements [MultiMedia](#).

5.13.3.6 set_duration()

```
void Video::set_duration (
    unsigned int duration ) [inline]
```

Sets the duration of the video.

Parameters

<i>duration</i>	The duration to set for the video in seconds.
-----------------	---

5.13.3.7 write()

```
void Video::write (
    ostream & f ) const [inline], [override], [virtual]
```

Writes the video's attributes to an output stream.

Parameters

<i>f</i>	The output stream to write to.
----------	--------------------------------

Reimplemented from [MultiMedia](#).

5.13.4 Friends And Related Function Documentation

5.13.4.1 Film

```
friend class Film [friend]
```


5.13.4.2 MediaManager

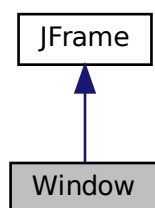
```
friend class MediaManager [friend]
```

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

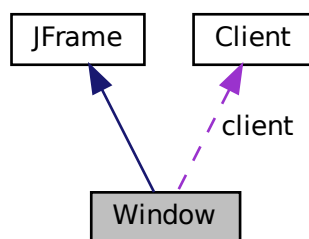
- [cpp/Video.h](#)

5.14 Window Class Reference

Inheritance diagram for Window:



Collaboration diagram for Window:



Classes

- class `Click1`
- class `Click2`
- class `Click3`
- class `Click4`
- class `Click5`
- class `Click6`
- class `Click7`
- class `Click8`
- class `Click9`

Public Member Functions

- [Window](#) ()

Static Public Member Functions

- static void [main](#) (String argv[])

5.14.1 Detailed Description

The main GUI window for the Media Player application. Provides a user interface to interact with the media server.

5.14.2 Constructor & Destructor Documentation

5.14.2.1 Window()

```
Window.Window ( ) [inline]
```

Constructs the main application window. Initializes the GUI components and sets up event handlers.

5.14.3 Member Function Documentation

5.14.3.1 main()

```
static void Window.main (
    String argv[] ) [inline], [static]
```

Main entry point for the application.

Parameters

<i>argv</i>	Command line arguments (not used)
-------------	-----------------------------------

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

- swing/[Window.java](#)

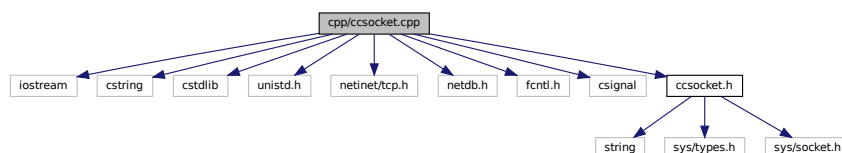
Chapter 6

File Documentation

6.1 cpp/ccsocket.cpp File Reference

```
#include <iostream>
#include <cstring>
#include <cstdlib>
#include <unistd.h>
#include <netinet/tcp.h>
#include <netdb.h>
#include <fcntl.h>
#include <csignal>
#include "ccsocket.h"
```

Include dependency graph for ccsocket.cpp:



Classes

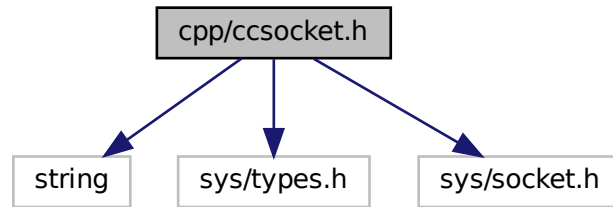
- struct [InputBuffer](#)

6.2 cpp/ccsocket.h File Reference

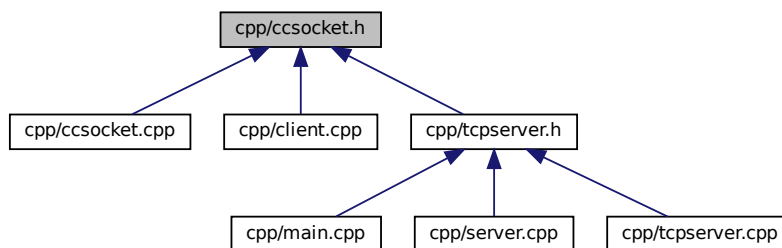
```
#include <string>
#include <sys/types.h>
```

```
#include <sys/socket.h>
```

Include dependency graph for ccsocket.h:



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



Classes

- class [Socket](#)
- class [ServerSocket](#)
- class [SocketBuffer](#)

Macros

- #define [SOCKET](#) int
- #define [SOCKADDR](#) struct sockaddr
- #define [SOCKADDR_IN](#) struct sockaddr_in
- #define [INVALID_SOCKET](#) -1
- #define [SOCKSIZE](#) ssize_t
- #define [SOCKDATA](#) void
- #define [NO_SIGPIPE](#)(flags) (flags)

6.2.1 Macro Definition Documentation

6.2.1.1 INVALID_SOCKET

```
#define INVALID_SOCKET -1
```

6.2.1.2 NO_SIGPIPE_

```
#define NO_SIGPIPE_(  
    flags ) (flags)
```

6.2.1.3 SOCKADDR

```
#define SOCKADDR struct sockaddr
```

6.2.1.4 SOCKADDR_IN

```
#define SOCKADDR_IN struct sockaddr_in
```

6.2.1.5 SOCKDATA

```
#define SOCKDATA void
```

6.2.1.6 SOCKET

```
#define SOCKET int
```

6.2.1.7 SOCKSIZE

```
#define SOCKSIZE ssize_t
```

6.3 ccsocket.h

[Go to the documentation of this file.](#)

```

1 //
2 //  ccsocket:  C++ Classes for TCP/IP and UDP Datagram INET Sockets.
3 //  (c) Eric Lecolinet 2016/2020 - https://www.telecom-paris.fr/~elc
4 //
5 //  - Socket:    TCP/IP or UDP/Datagram IPv4 socket
6 //  - ServerSocket:  TCP/IP Socket Server
7 //  - SocketBuffer:  preserves record boundaries when exchanging data
8 //                   between TCP/IP sockets.
9 //
10
11 #ifndef ccuty_ccsocket
12 #define ccuty_ccsocket 1
13
14 #include <string>
15
16 #if defined(_WIN32) || defined(_WIN64)
17 #include <winsock2.h>
18 #define SOCKSIZE int
19 #define SOCKDATA char
20 #else
21 #include <sys/types.h>
22 #include <sys/socket.h>
23 #define SOCKET int
24 #define SOCKADDR struct sockaddr
25 #define SOCKADDR_IN struct sockaddr_in
26 #define INVALID_SOCKET -1
27 #define SOCKSIZE ssize_t
28 #define SOCKDATA void
29 #endif
30
31 // ignore SIGPIPEs when possible
32 #if defined(MSG_NOSIGNAL)
33 #define NO_SIGPIPE_(flags) (flags | MSG_NOSIGNAL)
34 #else
35 #define NO_SIGPIPE_(flags) (flags)
36 #endif
37
38 class Socket {
39 public:
40     enum Errors { Failed = -1, InvalidSocket = -2, UnknownHost = -3 };
41
42     static void startup();
43     static void cleanup();
44
45     Socket(int type = SOCK_STREAM);
46     Socket(int type, SOCKET sockfd);
47     ~Socket();
48
49     int connect(const std::string& host, int port);
50     int bind(int port);
51     int bind(const std::string& host, int port);
52     int close();
53
54     bool isClosed() const { return sockfd_ == INVALID_SOCKET; }
55
56     SOCKET descriptor() { return sockfd_; }
57
58     void shutdownInput();
59     void shutdownOutput();
60
61     SOCKSIZE send(const SOCKDATA* buf, size_t len, int flags = 0) {
62         return ::send(sockfd_, buf, len, NO_SIGPIPE_(flags));
63     }
64
65     SOCKSIZE receive(SOCKDATA* buf, size_t len, int flags = 0) {
66         return ::recv(sockfd_, buf, len, flags);
67     }
68
69 #if !defined(_WIN32) && !defined(_WIN64)
70     SOCKSIZE sendTo(void const* buf, size_t len, int flags,
71                     SOCKADDR const* to, socklen_t addrlen) {
72         return ::sendto(sockfd_, buf, len, NO_SIGPIPE_(flags), to, addrlen);
73     }
74 #endif
75
76 private:
77     SOCKET sockfd_ = INVALID_SOCKET;
78 };

```

```

130     SOCKSIZE receiveFrom(void* buf, size_t len, int flags,
131                          SOCKADDR* from, socklen_t* addrlen) {
132         return ::recvfrom(sockfd_, buf, len, flags, from, addrlen);
133     }
134
135     int setReceiveBufferSize(int size);
136
137     int setReuseAddress(bool);
138
139     int setSendBufferSize(int size);
140
141     int setSoLinger(bool, int linger);
142
143     int setSoTimeout(int timeout);
144
145     int setTcpNoDelay(bool);
146
147     int getReceiveBufferSize() const;
148
149     bool getReuseAddress() const;
150
151     int getSendBufferSize() const;
152
153     bool getSoLinger(int& linger) const;
154
155     int getSoTimeout() const;
156
157     bool getTcpNoDelay() const;
158
159 #endif
160
161 private:
162     friend class ServerSocket;
163
164     // Initializes a local INET4 address, returns 0 on success, -1 otherwise.
165     int setLocalAddress(SOCKADDR_IN& addr, int port);
166     // Initializes a remote INET4 address, returns 0 on success, -1 otherwise.
167     int setAddress(SOCKADDR_IN& addr, const std::string& host, int port);
168
169     SOCKET sockfd_{};
170     Socket(const Socket&) = delete;
171     Socket& operator=(const Socket&) = delete;
172     Socket& operator=(Socket&&) = delete;
173 };
174
175 class ServerSocket {
176 public:
177     ServerSocket();
178
179     ~ServerSocket();
180
181     Socket* accept();
182
183     int bind(int port, int backlog = 50);
184
185     int close();
186
187     bool isClosed()const { return sockfd_ == INVALID_SOCKET; }
188
189     SOCKET descriptor() { return sockfd_; }
190
191 #if !defined(_WIN32) && !defined(_WIN64)
192     int setReceiveBufferSize(int size);
193
194     int setReuseAddress(bool);
195
196     int setSoTimeout(int timeout);
197
198     int setTcpNoDelay(bool);
199 #endif
200
201 private:
202     Socket* createSocket(SOCKET);
203     SOCKET sockfd_{}; // listening socket.
204     ServerSocket(const ServerSocket&) = delete;
205     ServerSocket& operator=(const ServerSocket&) = delete;
206     ServerSocket& operator=(ServerSocket&&) = delete;
207 };
208
209 class SocketBuffer {
210 public:
211     SocketBuffer(Socket*, size_t inputSize = 8192, size_t outputSize = 8192);

```

```

284  SocketBuffer(Socket&, size_t inputSize = 8192, size_t ouputSize = 8192);
286
287  ~SocketBuffer();
288
289  SOCKSIZE readLine(std::string& message);
290
291  SOCKSIZE writeLine(const std::string& message);
292
293  SOCKSIZE read(char* buffer, size_t len);
294
295  SOCKSIZE write(const char* str, size_t len);
296
297  Socket* socket() { return sock_; }
298
299  void setReadSeparator(int separ);
300  int readSeparator()const { return insep_; }
301  // @}
302
303  void setWriteSeparator(int separ);
304  int writeSeparator()const { return outsep_; }
305  // @}
306
307 private:
308  SocketBuffer(const SocketBuffer&) = delete;
309  SocketBuffer& operator=(const SocketBuffer&) = delete;
310  SocketBuffer& operator=(SocketBuffer&) = delete;
311
312 protected:
313  bool retrieveLine(std::string& str, SOCKSIZE received);
314  size_t insize_{}, outsize_{};
315  int insep_{}, outsep_{};
316  Socket* sock_{};
317  struct InputBuffer* in_{};
318 };
319
320 #endif

```

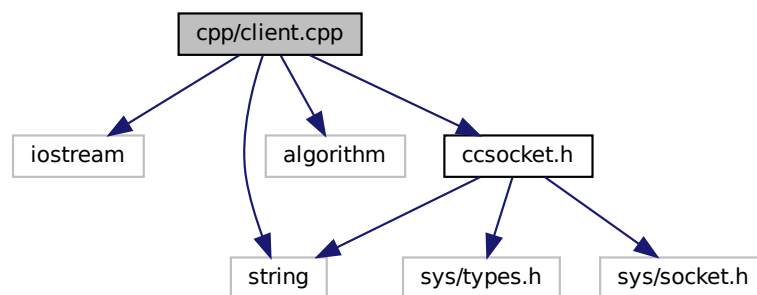
6.4 cpp/client.cpp File Reference

```

#include <iostream>
#include <string>
#include <algorithm>
#include "ccsocket.h"

```

Include dependency graph for client.cpp:



Functions

- int `main` ()

6.4.1 Function Documentation

6.4.1.1 main()

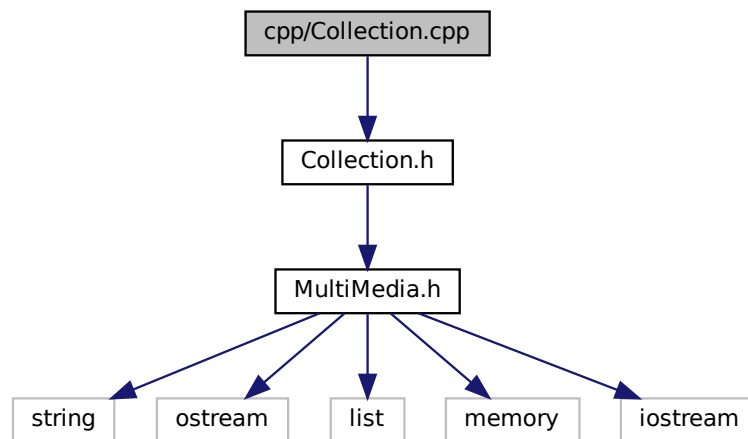
```
int main ( )
```

Lit une requete depuis le Terminal, envoie cette requete au serveur, recupere sa reponse et l'affiche sur le Terminal.
Noter que le programme bloque si le serveur ne repond pas.

6.5 cpp/Collection.cpp File Reference

```
#include "Collection.h"
```

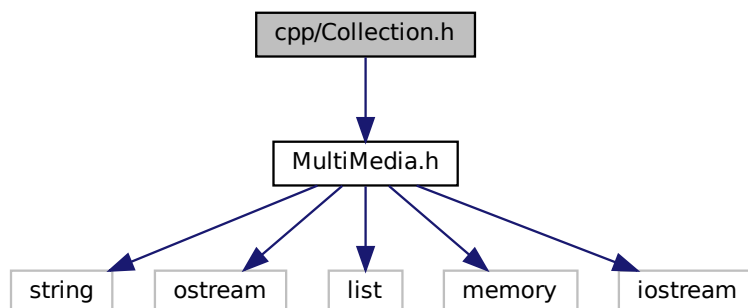
Include dependency graph for Collection.cpp:



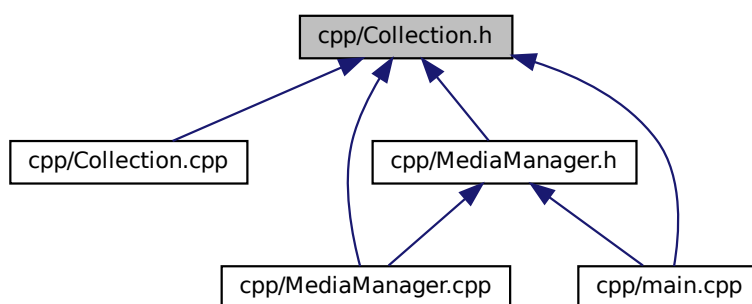
6.6 cpp/Collection.h File Reference

```
#include "MultiMedia.h"
```

Include dependency graph for Collection.h:



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



Classes

- class [Collection](#)
A collection of multimedia objects.

Typedefs

- typedef `std::shared_ptr< MultiMedia >` [MediaPtr](#)
Shared pointer to a [MultiMedia](#) object.

6.6.1 Typedef Documentation

6.6.1.1 MediaPtr

MediaPtr

Shared pointer to a [MultiMedia](#) object.

6.7 Collection.h

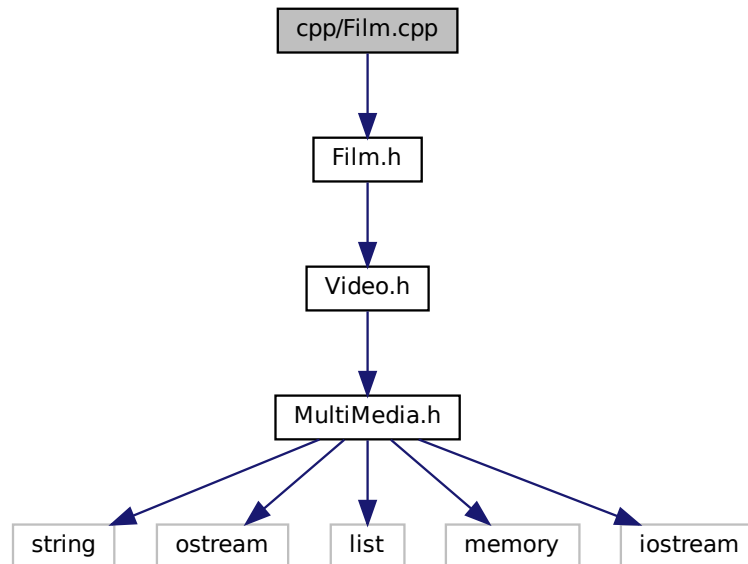
[Go to the documentation of this file.](#)

```
1 #ifndef COLLECTION_H
2 #define COLLECTION_H
3 #include "MultiMedia.h"
4
9 typedef std::shared_ptr<MultiMedia> MediaPtr;
10
19 class Collection : public list<MediaPtr>
20 {
21     friend class MediaManager;
22 private:
23     string name {};
24
29     Collection(string name);
30
34     Collection(){};
35
36 public:
40     ~Collection(){};
41
46     string get_name() const;
47
52     string get_class_name() const;
53
58     void write(ostream& f) const;
59
64     void read(istream& f);
65
70     void disp(ostream& out) const;
71 };
72
73 #endif
```

6.8 cpp/Film.cpp File Reference

```
#include "Film.h"
```

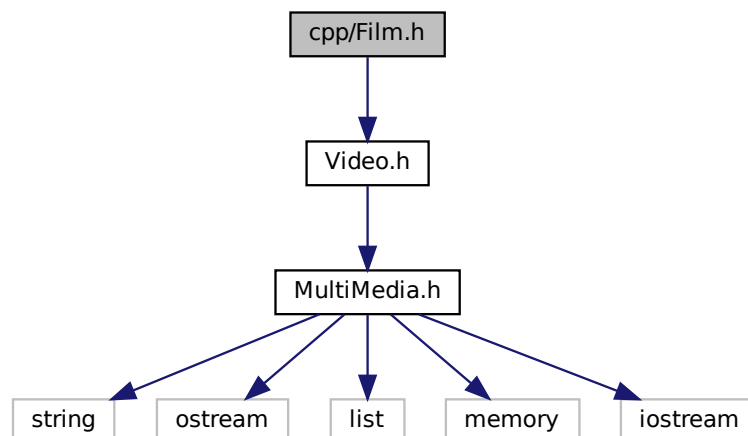
Include dependency graph for Film.cpp:



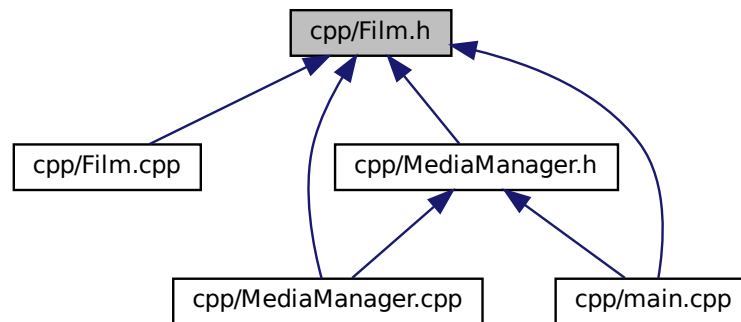
6.9 cpp/Film.h File Reference

```
#include "Video.h"
```

Include dependency graph for Film.h:



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



Classes

- class [Film](#)

A class representing a film with chapters, derived from [Video](#).

6.10 Film.h

[Go to the documentation of this file.](#)

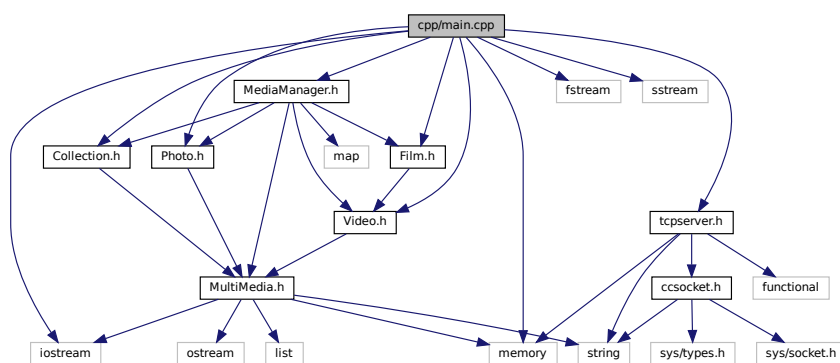
```

1  #ifndef FILM_H
2  #define FILM_H
3  #include "Video.h"
4
12 class Film : public Video
13 {
14     friend class MediaManager;
15 private:
16     unsigned int* chapters {};
17     unsigned int nb_chapters {};
18
22     Film() {};
23
33     Film(string name, string path, unsigned int duration,
34           unsigned int const* chapters, unsigned int nb_chapters);
35
40     Film(const Film& from);
41
42 public:
48     Film& operator=(const Film& from);
49
53     ~Film();
54
60     void set_chapters(unsigned int const* chapters, unsigned int nb_chapters);
61
66     const unsigned int* get_chapters() const;
67
72     unsigned int get_nb_chapters() const;
73
78     void disp(ostream& out) const override;
79
84     string get_class_name() const override;
85
90     void serial_chapters(ostream& f) const;
91
96     void deserial_chapters(istream& f);
97
102    void write(ostream& f) const override;
103
108    void read(istream& f) override;
109 };
110
111 #endif
  
```

6.11 cpp/main.cpp File Reference

```
#include <iostream>
#include <memory>
#include "Video.h"
#include "Photo.h"
#include "Film.h"
#include "Collection.h"
#include "MediaManager.h"
#include <fstream>
#include <sstream>
#include "tcpserver.h"
```

Include dependency graph for main.cpp:



Macros

- `#define SERVER_v`

Functions

- `int main (int argc, char *argv[])`
Main function for server mode of the multimedia application.

Variables

- `const int PORT = 3331`

6.11.1 Macro Definition Documentation

6.11.1.1 SERVER_v

```
#define SERVER_v
```

6.11.2 Function Documentation

6.11.2.1 main()

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

Main function for server mode of the multimedia application.

Creates a [MediaManager](#) with sample data and starts a TCP server to handle client requests.

Parameters

<i>argc</i>	Number of command-line arguments.
<i>argv</i>	Array of command-line arguments.

Returns

int Returns 0 on success, 1 on server startup failure.

6.11.3 Variable Documentation

6.11.3.1 PORT

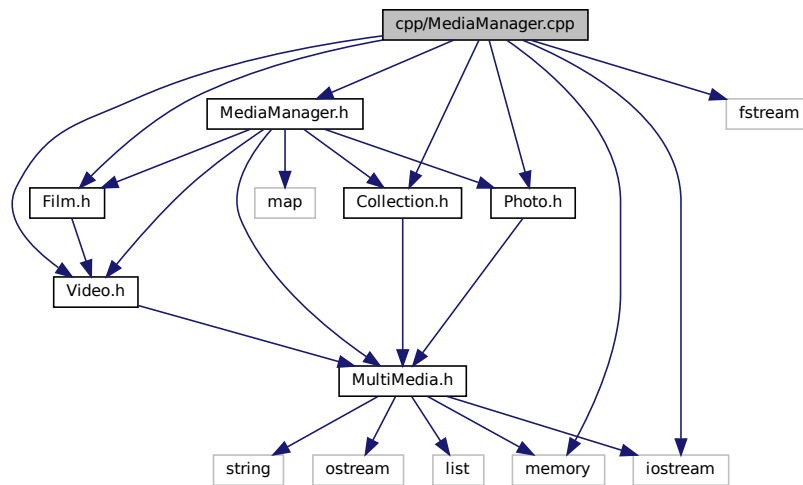
```
const int PORT = 3331
```

6.12 cpp/MediaManager.cpp File Reference

```
#include "MediaManager.h"
#include <fstream>
#include <iostream>
#include <memory>
#include "Video.h"
#include "Photo.h"
#include "Film.h"
```

```
#include "Collection.h"
```

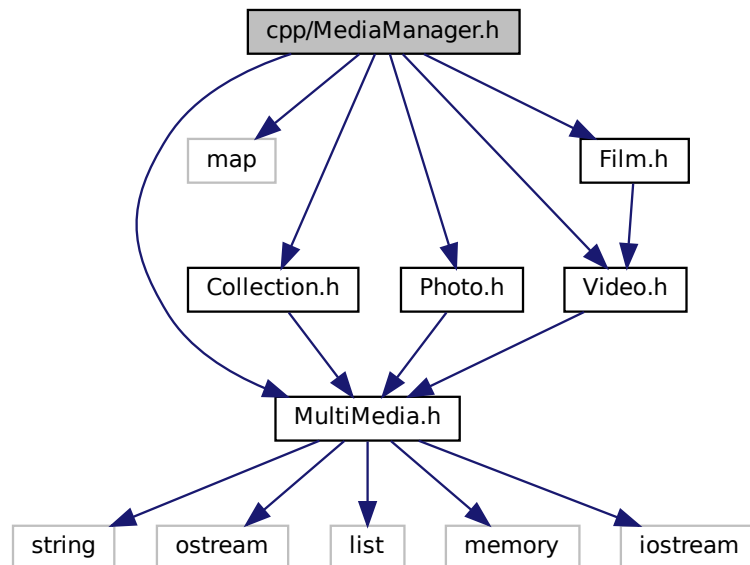
Include dependency graph for MediaManager.cpp:



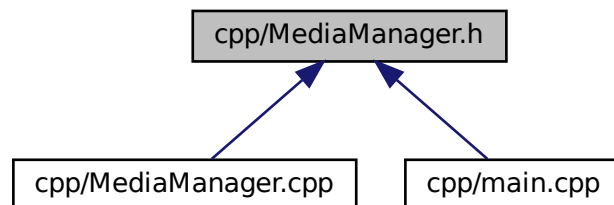
6.13 cpp/MediaManager.h File Reference

```
#include "MultiMedia.h"
#include <map>
#include "Collection.h"
#include "Photo.h"
#include "Video.h"
#include "Film.h"
```


Include dependency graph for MediaManager.h:



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



Classes

- class [MediaManager](#)
Manages collections of multimedia objects and their operations.

Typedefs

- typedef `std::shared_ptr< MultiMedia >` [MediaPtr](#)
- typedef `std::shared_ptr< Collection >` [CollectPtr](#)
Shared pointer to a [Collection](#) object.

- `typedef std::shared_ptr< Photo > SharPhotoPtr`
Shared pointer to a [Photo](#) object.
- `typedef std::shared_ptr< Video > SharVideoPtr`
Shared pointer to a [Video](#) object.
- `typedef std::shared_ptr< Film > SharFilmPtr`
Shared pointer to a [Film](#) object.
- `typedef std::shared_ptr< Collection > SharCollectPtr`
Shared pointer to a [Collection](#) object.
- `typedef std::map< string, CollectPtr > Dict_collection`
Map of collection names to [Collection](#) shared pointers.
- `typedef std::map< string, MediaPtr > Dict_Media`
Map of media names to [MultiMedia](#) shared pointers.

6.13.1 Typedef Documentation

6.13.1.1 CollectPtr

[CollectPtr](#)

Shared pointer to a [Collection](#) object.

6.13.1.2 Dict_collection

[Dict_collection](#)

Map of collection names to [Collection](#) shared pointers.

6.13.1.3 Dict_Media

[Dict_Media](#)

Map of media names to [MultiMedia](#) shared pointers.

6.13.1.4 MediaPtr

```
typedef std::shared_ptr<MultiMedia> MediaPtr
```

6.13.1.5 SharCollectPtr

`SharCollectPtr`

Shared pointer to a [Collection](#) object.

6.13.1.6 SharFilmPtr

`SharFilmPtr`

Shared pointer to a [Film](#) object.

6.13.1.7 SharPhotoPtr

`SharPhotoPtr`

Shared pointer to a [Photo](#) object.

6.13.1.8 SharVideoPtr

`SharVideoPtr`

Shared pointer to a [Video](#) object.

6.14 MediaManager.h

[Go to the documentation of this file.](#)

```
1 #ifndef MEDIA_MANAGER_H
2 #define MEDIA_MANAGER_H
3 #include "MultiMedia.h"
4 #include <map>
5 #include "Collection.h"
6 #include "Photo.h"
7 #include "Video.h"
8 #include "Film.h"
9
14 typedef std::shared_ptr<MultiMedia> MediaPtr;
15
20 typedef std::shared_ptr<Collection> CollectPtr;
21
26 typedef std::shared_ptr<Photo> SharPhotoPtr;
27
32 typedef std::shared_ptr<Video> SharVideoPtr;
33
38 typedef std::shared_ptr<Film> SharFilmPtr;
39
44 typedef std::shared_ptr<Collection> SharCollectPtr;
45
50 typedef std::map<string, CollectPtr> Dict_collection;
51
56 typedef std::map<string, MediaPtr> Dict_Media;
57
65 class MediaManager
```

```

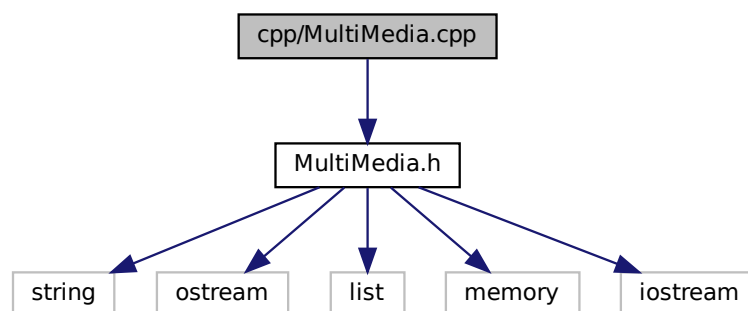
66 {
67 private:
68     Dict_collection groups {};
69     Dict_Media medias {};
70
71 public:
72     MediaManager(){};
73
74     ~MediaManager(){};
75
76     SharPhotoPtr create_photo(string name, string path, double width, double height);
77
78     SharVideoPtr create_video(string name, string path, unsigned int duration);
79
80     SharFilmPtr create_Film(string name, string path, unsigned int duration,
81                             unsigned int const* chapters, unsigned int nb_chapters);
82
83     SharCollectPtr create_collection(string name);
84
85     void disp_media(ostream& out, string name) const;
86
87     void disp_collection(ostream& out, string name) const;
88
89     void play_media(string name) const;
90
91     void delete_media(string name);
92
93     void delete_collection(string name);
94
95     void disp_all(ostream& out) const;
96
97     void write(ostream& f) const;
98
99     void read(istream& f);
100 };
101 #endif

```

6.15 cpp/MultiMedia.cpp File Reference

```
#include "MultiMedia.h"
```

Include dependency graph for MultiMedia.cpp:



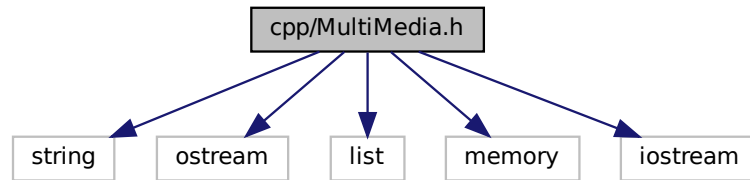
6.16 cpp/MultiMedia.h File Reference

```

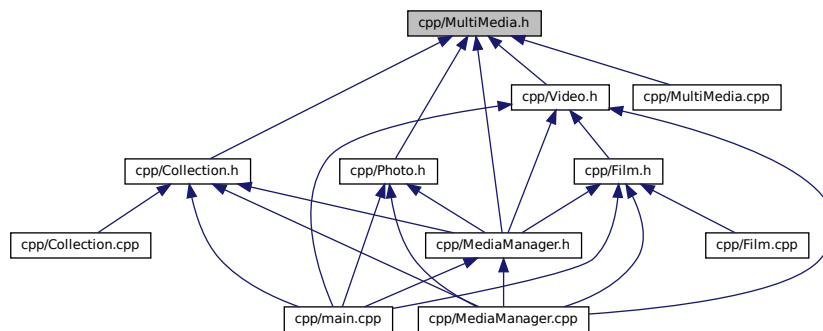
#include <string>
#include <ostream>
#include <list>

```

```
#include <memory>
#include <iostream>
Include dependency graph for MultiMedia.h:
```



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



Classes

- class [MultiMedia](#)
Represents a multimedia file.

6.17 MultiMedia.h

[Go to the documentation of this file.](#)

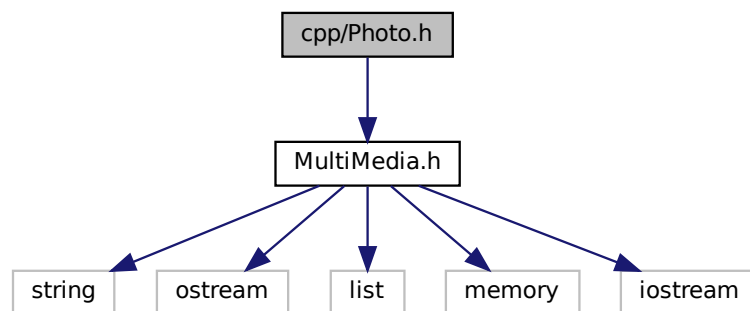
```
1 #ifndef MULTI_MEDIA_H
2 #define MULTI_MEDIA_H
3
4 #include <string>
5 #include <ostream>
6 #include <list>
7 #include <memory>
8 #include <iostream>
9 using namespace std;
10
11
12
13
14
15
16
17
18 class MultiMedia
19 {
20 private:
21     string name{};
22     string path{};
23 }
```

```
24 public:
25     MultiMedia();
26
27     MultiMedia(string name, string path);
28
29     ~MultiMedia();
30
31     string get_name() const;
32
33     string get_path() const;
34
35     void set_name(string name);
36
37     void set_path(string path);
38
39     virtual void disp(ostream &out) const;
40
41     virtual void run() const = 0;
42     virtual string get_class_name() const = 0;
43
44     virtual void write(ostream & f) const ;
45     virtual void read(istream & f);
46 };
47
48 #endif // MULTI_MEDIA_
```

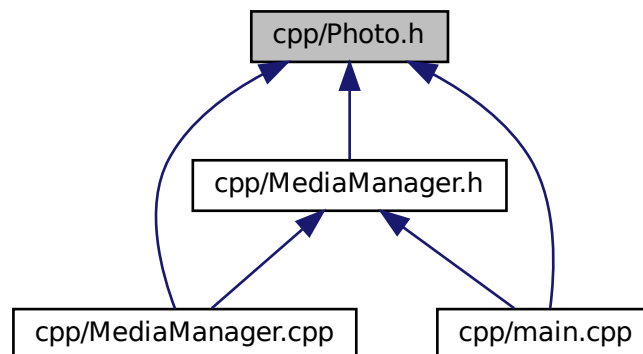
6.18 cpp/Photo.h File Reference

#include "MultiMedia.h"

Include dependency graph for Photo.h:



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



Classes

- class [Photo](#)

A class representing a photo file, derived from the [MultiMedia](#) base class.

6.19 Photo.h

[Go to the documentation of this file.](#)

```

1 #ifndef PHOTO_H
2 #define PHOTO_H
3 #include "MultiMedia.h"
4
12 class Photo : public MultiMedia
13 {
14     friend class MediaManager;
15 private:
16     double width {};
17     double height {};
18
23     Photo() {}
24
33     Photo(string name, string path, double width, double height) :
34     MultiMedia(name, path, width{width}, height{height}) {}
35
36
37 public:
42     ~Photo() {
43         cout<< " A Photo classe has been destroyed U Monster"<<endl;
44     };
45
51     void set_width(double width) {
52         this->width = width;
53     }
54
60     double get_width()const {
61         return this->width;
62     }
63
69     void set_height(double height) {
70         this->height = height;
71     }
72
78     double get_height()const {
79         return this->height;
80     }
  
```

```

81
89     void disp(ostream & out) const override {
90         MultiMedia::disp(out);
91         out << "\nwidth : " << this->get_width() <<
92             "\nheight : " << this->get_height() << endl;
93     }
94
101    void run() const override {
102        string command = "mpv --keep-open";
103        command = command + " " + this->get_path() + " &";
104        system(command.data());
105    }
106
112    string get_class_name() const override{
113        return "Photo";
114    }
115
121    void write(ostream & f) const override{
122        f << "Photo" << '\n';
123        MultiMedia::write(f);
124        f << width << '\n' << height << endl ;
125    }
126
132    void read(istream & f){
133        MultiMedia::read(f);
134        f >> width >> height;
135    }
136 };
137
138 #endif

```

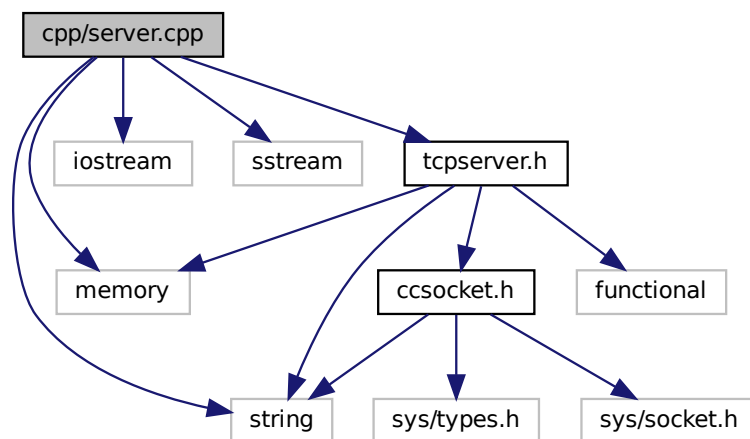
6.20 cpp/server.cpp File Reference

```

#include <memory>
#include <string>
#include <iostream>
#include <sstream>
#include "tcpserver.h"

```

Include dependency graph for server.cpp:



Functions

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

Variables

- const int `PORT` = 3331

6.20.1 Function Documentation

6.20.1.1 main()

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

6.20.2 Variable Documentation

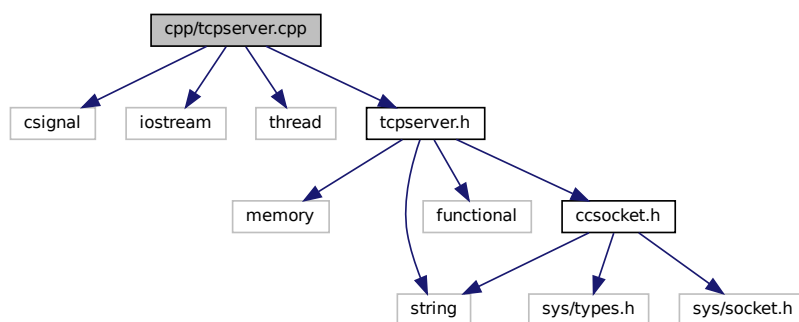
6.20.2.1 PORT

```
const int PORT = 3331
```

6.21 cpp/tcpserver.cpp File Reference

```
#include <csignal>  
#include <iostream>  
#include <thread>  
#include "tcpserver.h"
```

Include dependency graph for tcpserver.cpp:



Classes

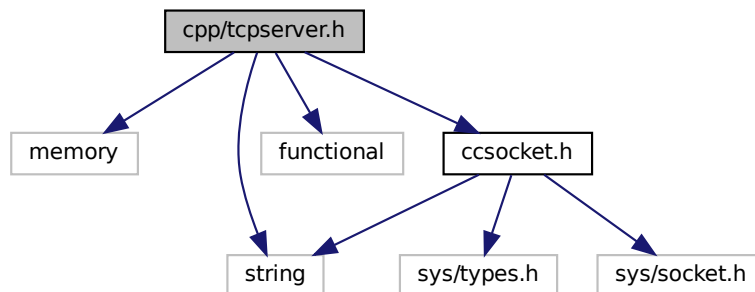
- class [SocketCnx](#)

Connection with a given client. Each [SocketCnx](#) uses a different thread.

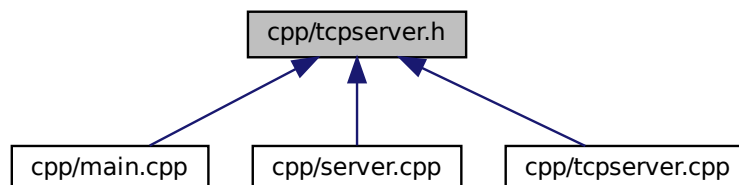
6.22 cpp/tcpserver.h File Reference

```
#include <memory>
#include <string>
#include <functional>
#include "ccsocket.h"
```

Include dependency graph for tcpserver.h:



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



Classes

- class [TCPServer](#)

6.23 tcpserver.h

[Go to the documentation of this file.](#)

```

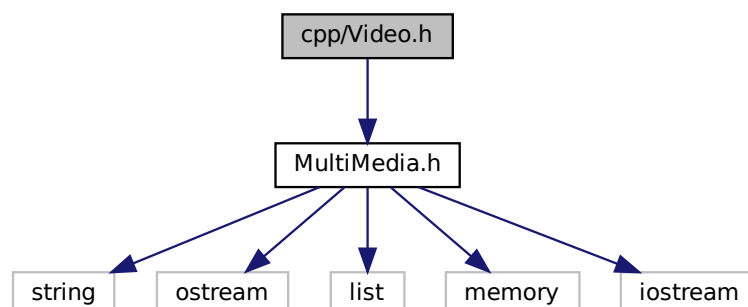
1 //
2 //  tcpserver:  TCP/IP INET Server.
3 //  (c) Eric Lecolinet - Telecom ParisTech - 2016.
4 //  http://www.telecom-paristech.fr/~elc
5 //
6
7 #ifndef __tcpserver__
8 #define __tcpserver__
9 #include <memory>
10 #include <string>
11 #include <functional>
12 #include "ccsocket.h"
13
14 class TCPConnection;
15 class TCPLock;
16
17 class TCPServer {
18 public:
19     using Callback =
20         std::function< bool(std::string const& request, std::string& response) >;
21
22     TCPServer(Callback const& callback);
23
24     virtual ~TCPServer();
25
26     virtual int run(int port);
27 private:
28     friend class TCPLock;
29     friend class SocketCnx;
30
31     TCPServer(TCPServer const&) = delete;
32     TCPServer& operator=(TCPServer const&) = delete;
33     void error(std::string const& msg);
34
35     ServerSocket servsock_;
36     Callback callback_{};
37 };
38 #endif

```

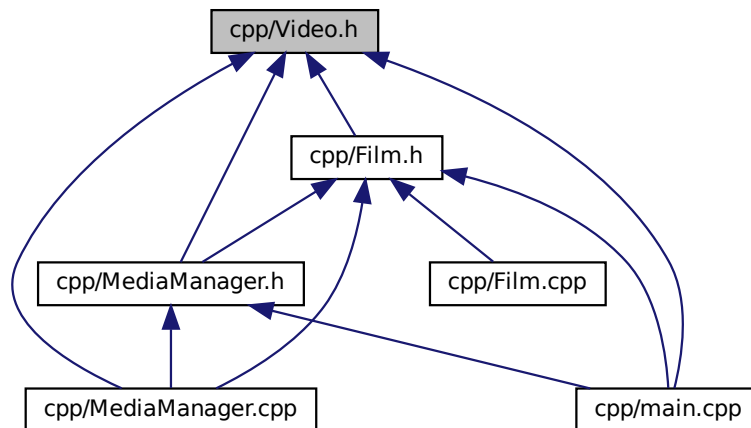
6.24 cpp/Video.h File Reference

```
#include "MultiMedia.h"
```

Include dependency graph for Video.h:



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



Classes

- class [Video](#)

A class representing a video file, derived from the [MultiMedia](#) base class.

6.25 Video.h

[Go to the documentation of this file.](#)

```

1 #ifndef VIDEO_H
2 #define VIDEO_H
3 #include "MultiMedia.h"
4
12 class Video : public MultiMedia
13 {
14     friend class MediaManager;
15     friend class Film;
16 private:
17     unsigned int duration {};
18
23     Video() {}
24
32     Video(string name, string path, unsigned int duration) :
33     MultiMedia(name, path), duration{duration} {}
34
35 public:
39     ~Video() {
40         cout << " A Video classe has been destroyed U Monster" << endl;
41     };
42
48     void set_duration(unsigned int duration) {
49         this->duration = duration ;
50     }
51
57     unsigned int get_duration() const {
58         return this->duration ;
59     }
60
68     void disp(ostream & out) const override {
69         MultiMedia::disp(out);
70         out << "\nduration : " << this->get_duration() << endl;
71     }
72
79     void run() const override {

```

```
80         string command = "mpv --keep-open";
81         command = command + " " + this->get_path() + " &";
82         system(command.data());
83     }
84
85     string get_class_name()const override{
86         return "Video";
87     }
88
89     void write(ostream & f)const override{
90         f << "Video" << '\n';
91         MultiMedia::write(f);
92         f << duration << '\n' ;
93     }
94
95     void read(istream & f){
96         MultiMedia::read(f);
97         f >> duration ;
98     }
99 };
100 #endif
```

6.26 README.md File Reference

6.27 swing/Client.java File Reference

Classes

- class [Client](#)

6.28 swing/Window.java File Reference

Classes

- class [Window](#)
- class [Window.Click1](#)
- class [Window.Click2](#)
- class [Window.Click3](#)
- class [Window.Click4](#)
- class [Window.Click5](#)
- class [Window.Click6](#)
- class [Window.Click7](#)
- class [Window.Click8](#)
- class [Window.Click9](#)

Index

- ~Collection
 - Collection, [13](#)
- ~Film
 - Film, [17](#)
- ~InputBuffer
 - InputBuffer, [22](#)
- ~MediaManager
 - MediaManager, [23](#)
- ~MultiMedia
 - MultiMedia, [30](#)
- ~Photo
 - Photo, [35](#)
- ~ServerSocket
 - ServerSocket, [38](#)
- ~Socket
 - Socket, [43](#)
- ~SocketBuffer
 - SocketBuffer, [51](#)
- ~SocketCnx
 - SocketCnx, [56](#)
- ~TCPServer
 - TCPServer, [58](#)
- ~Video
 - Video, [60](#)
- accept
 - ServerSocket, [39](#)
- begin
 - InputBuffer, [22](#)
- bind
 - ServerSocket, [39](#)
 - Socket, [43](#), [44](#)
- buffer
 - InputBuffer, [22](#)
- Callback
 - TCPServer, [57](#)
- ccsocket.h
 - INVALID_SOCKET, [66](#)
 - NO_SIGPIPE_, [67](#)
 - SOCKADDR, [67](#)
 - SOCKADDR_IN, [67](#)
 - SOCKDATA, [67](#)
 - SOCKET, [67](#)
 - SOCKSIZE, [67](#)
- cleanup
 - Socket, [44](#)
- Client, [11](#)
 - Client, [11](#)
 - main, [11](#)
 - send, [11](#)
- client.cpp
 - main, [71](#)
- close
 - ServerSocket, [39](#)
 - Socket, [44](#)
- Collection, [12](#)
 - ~Collection, [13](#)
 - disp, [13](#)
 - get_class_name, [14](#)
 - get_name, [14](#)
 - MediaManager, [15](#)
 - read, [14](#)
 - write, [15](#)
- Collection.h
 - MediaPtr, [72](#)
- CollectPtr
 - MediaManager.h, [80](#)
- connect
 - Socket, [44](#)
- cpp/ccsocket.cpp, [65](#)
- cpp/ccsocket.h, [65](#), [68](#)
- cpp/client.cpp, [70](#)
- cpp/Collection.cpp, [71](#)
- cpp/Collection.h, [71](#), [73](#)
- cpp/Film.cpp, [74](#)
- cpp/Film.h, [74](#), [75](#)
- cpp/main.cpp, [76](#)
- cpp/MediaManager.cpp, [77](#)
- cpp/MediaManager.h, [78](#), [81](#)
- cpp/MultiMedia.cpp, [82](#)
- cpp/MultiMedia.h, [82](#), [83](#)
- cpp/Photo.h, [84](#), [85](#)
- cpp/server.cpp, [86](#)
- cpp/tcpserver.cpp, [87](#)
- cpp/tcpserver.h, [88](#), [89](#)
- cpp/Video.h, [89](#), [90](#)
- create_collection
 - MediaManager, [24](#)
- create_Film
 - MediaManager, [24](#)
- create_photo
 - MediaManager, [25](#)
- create_video
 - MediaManager, [25](#)
- delete_collection
 - MediaManager, [26](#)
- delete_media

- MediaManager, 26
- descriptor
 - ServerSocket, 39
 - Socket, 44
- deserial_chapters
 - Film, 17
- Dict_collection
 - MediaManager.h, 80
- Dict_Media
 - MediaManager.h, 80
- disp
 - Collection, 13
 - Film, 18
 - MultiMedia, 30
 - Photo, 35
 - Video, 60
- disp_all
 - MediaManager, 26
- disp_collection
 - MediaManager, 27
- disp_media
 - MediaManager, 27
- end
 - InputBuffer, 22
- Errors
 - Socket, 42
- Failed
 - Socket, 43
- Film, 16
 - ~Film, 17
 - deserial_chapters, 17
 - disp, 18
 - get_chapters, 18
 - get_class_name, 18
 - get_nb_chapters, 18
 - MediaManager, 21
 - operator=, 19
 - read, 19
 - serial_chapters, 20
 - set_chapters, 20
 - Video, 62
 - write, 20
- get_chapters
 - Film, 18
- get_class_name
 - Collection, 14
 - Film, 18
 - MultiMedia, 31
 - Photo, 35
 - Video, 61
- get_duration
 - Video, 61
- get_height
 - Photo, 35
- get_name
 - Collection, 14
- MultiMedia, 31
- get_nb_chapters
 - Film, 18
- get_path
 - MultiMedia, 31
- get_width
 - Photo, 36
- getReceiveBufferSize
 - Socket, 45
- getReuseAddress
 - Socket, 45
- getSendBufferSize
 - Socket, 45
- getSoLinger
 - Socket, 45
- getSoTimeout
 - Socket, 45
- getTcpNoDelay
 - Socket, 45
- in_
 - SocketBuffer, 54
- InputBuffer, 21
 - ~InputBuffer, 22
 - begin, 22
 - buffer, 22
 - end, 22
 - InputBuffer, 21
 - remaining, 22
- insep_
 - SocketBuffer, 54
- insize_
 - SocketBuffer, 54
- INVALID_SOCKET
 - ccsocket.h, 66
- InvalidSocket
 - Socket, 43
- isClosed
 - ServerSocket, 39
 - Socket, 46
- main
 - Client, 11
 - client.cpp, 71
 - main.cpp, 77
 - server.cpp, 87
 - Window, 64
- main.cpp
 - main, 77
 - PORT, 77
 - SERVER_v, 76
- MediaManager, 22
 - ~MediaManager, 23
 - Collection, 15
 - create_collection, 24
 - create_Film, 24
 - create_photo, 25
 - create_video, 25
 - delete_collection, 26

- delete_media, 26
- disp_all, 26
- disp_collection, 27
- disp_media, 27
- Film, 21
- MediaManager, 23
- Photo, 37
- play_media, 27
- read, 28
- Video, 62
- write, 28
- MediaManager.h
 - CollectPtr, 80
 - Dict_collection, 80
 - Dict_Media, 80
 - MediaPtr, 80
 - SharCollectPtr, 80
 - SharFilmPtr, 81
 - SharPhotoPtr, 81
 - SharVideoPtr, 81
- MediaPtr
 - Collection.h, 72
 - MediaManager.h, 80
- MultiMedia, 28
 - ~MultiMedia, 30
 - disp, 30
 - get_class_name, 31
 - get_name, 31
 - get_path, 31
 - MultiMedia, 30
 - read, 31
 - run, 32
 - set_name, 32
 - set_path, 32
 - write, 33
- NO_SIGPIPE_
 - ccsocket.h, 67
- operator=
 - Film, 19
- outsep_
 - SocketBuffer, 54
- outsize_
 - SocketBuffer, 54
- Photo, 33
 - ~Photo, 35
 - disp, 35
 - get_class_name, 35
 - get_height, 35
 - get_width, 36
 - MediaManager, 37
 - read, 36
 - run, 36
 - set_height, 36
 - set_width, 37
 - write, 37
- play_media
 - MediaManager, 27
- PORT
 - main.cpp, 77
 - server.cpp, 87
- processRequests
 - SocketCnx, 56
- read
 - Collection, 14
 - Film, 19
 - MediaManager, 28
 - MultiMedia, 31
 - Photo, 36
 - SocketBuffer, 51
 - Video, 61
- readLine
 - SocketBuffer, 51
- README.md, 91
- readSeparator
 - SocketBuffer, 52
- receive
 - Socket, 46
- receiveFrom
 - Socket, 46
- remaining
 - InputBuffer, 22
- retrieveLine
 - SocketBuffer, 52
- run
 - MultiMedia, 32
 - Photo, 36
 - TCPServer, 58
 - Video, 61
- send
 - Client, 11
 - Socket, 46
- sendTo
 - Socket, 47
- serial_chapters
 - Film, 20
- server.cpp
 - main, 87
 - PORT, 87
- server_
 - SocketCnx, 56
- SERVER_v
 - main.cpp, 76
- ServerSocket, 38
 - ~ServerSocket, 38
 - accept, 39
 - bind, 39
 - close, 39
 - descriptor, 39
 - isClosed, 39
 - ServerSocket, 38
 - setReceiveBufferSize, 40
 - setReuseAddress, 40
 - setSoTimeout, 40

- setTcpNoDelay, 40
 - Socket, 49
- set_chapters
 - Film, 20
- set_duration
 - Video, 62
- set_height
 - Photo, 36
- set_name
 - MultiMedia, 32
- set_path
 - MultiMedia, 32
- set_width
 - Photo, 37
- setReadSeparator
 - SocketBuffer, 52
- setReceiveBufferSize
 - ServerSocket, 40
 - Socket, 47
- setReuseAddress
 - ServerSocket, 40
 - Socket, 47
- setSendBufferSize
 - Socket, 47
- setSoLinger
 - Socket, 48
- setSoTimeout
 - ServerSocket, 40
 - Socket, 48
- setTcpNoDelay
 - ServerSocket, 40
 - Socket, 48
- setWriteSeparator
 - SocketBuffer, 52
- SharCollectPtr
 - MediaManager.h, 80
- SharFilmPtr
 - MediaManager.h, 81
- SharPhotoPtr
 - MediaManager.h, 81
- SharVideoPtr
 - MediaManager.h, 81
- shutdownInput
 - Socket, 48
- shutdownOutput
 - Socket, 48
- sock_
 - SocketBuffer, 54
 - SocketCnx, 56
- SOCKADDR
 - ccsocket.h, 67
- SOCKADDR_IN
 - ccsocket.h, 67
- sockbuf_
 - SocketCnx, 56
- SOCKDATA
 - ccsocket.h, 67
- SOCKET
 - ccsocket.h, 67
- Socket, 41
 - ~Socket, 43
 - bind, 43, 44
 - cleanup, 44
 - close, 44
 - connect, 44
 - descriptor, 44
 - Errors, 42
 - Failed, 43
 - getReceiveBufferSize, 45
 - getReuseAddress, 45
 - getSendBufferSize, 45
 - getSoLinger, 45
 - getSoTimeout, 45
 - getTcpNoDelay, 45
 - InvalidSocket, 43
 - isClosed, 46
 - receive, 46
 - receiveFrom, 46
 - send, 46
 - sendTo, 47
 - ServerSocket, 49
 - setReceiveBufferSize, 47
 - setReuseAddress, 47
 - setSendBufferSize, 47
 - setSoLinger, 48
 - setSoTimeout, 48
 - setTcpNoDelay, 48
 - shutdownInput, 48
 - shutdownOutput, 48
 - Socket, 43
 - startup, 48
 - UnknownHost, 43
- socket
 - SocketBuffer, 53
- SocketBuffer, 49
 - ~SocketBuffer, 51
 - in_, 54
 - insep_, 54
 - insize_, 54
 - outsep_, 54
 - outsize_, 54
 - read, 51
 - readLine, 51
 - readSeparator, 52
 - retrieveLine, 52
 - setReadSeparator, 52
 - setWriteSeparator, 52
 - sock_, 54
 - socket, 53
 - SocketBuffer, 51
 - write, 53
 - writeLine, 53
 - writeSeparator, 53
- SocketCnx, 55
 - ~SocketCnx, 56
 - processRequests, 56

- server_, [56](#)
- sock_, [56](#)
- sockbuf_, [56](#)
- SocketCnx, [55](#)
- TCPServer, [58](#)
- thread_, [56](#)
- SOCKSIZE
 - ccsocket.h, [67](#)
- startup
 - Socket, [48](#)
- swing/Client.java, [91](#)
- swing/Window.java, [91](#)
- TCPLock
 - TCPServer, [58](#)
- TCPServer, [57](#)
 - ~TCPServer, [58](#)
 - Callback, [57](#)
 - run, [58](#)
 - SocketCnx, [58](#)
 - TCPLock, [58](#)
 - TCPServer, [57](#)
- thread_
 - SocketCnx, [56](#)
- UnknownHost
 - Socket, [43](#)
- Video, [59](#)
 - ~Video, [60](#)
 - disp, [60](#)
 - Film, [62](#)
 - get_class_name, [61](#)
 - get_duration, [61](#)
 - MediaManager, [62](#)
 - read, [61](#)
 - run, [61](#)
 - set_duration, [62](#)
 - write, [62](#)
- Window, [63](#)
 - main, [64](#)
 - Window, [64](#)
- write
 - Collection, [15](#)
 - Film, [20](#)
 - MediaManager, [28](#)
 - MultiMedia, [33](#)
 - Photo, [37](#)
 - SocketBuffer, [53](#)
 - Video, [62](#)
- writeLine
 - SocketBuffer, [53](#)
- writeSeparator
 - SocketBuffer, [53](#)