Mutimedia

1.0

Generated by Doxygen 1.9.1

Chapter 1

README

1.1 Partie CPP

1.1.1 Task 4 | Photos et Videos

```
Comment appelle-t'on ce type de méthode et comment faut-il les déclarer?
```

Réponse:

Ce type de méthode s'appelle méthode pure virtuelle. Il fait déclarer avec le mot clé `virtual` et `=0`(pure specifier) à la fin. Par example `virtual f()=0;`

```
Si vous avez fait ce qui précède comme demandé, il ne sera plus possible
d'instancer des objets de la classe de base.  Pourquoi ?
```

Réponse:

Non, il n'est plus possible parce que on ne peut pas instancer d'un abstract class (au moin un de ses méthod virtual n'est pas implémenté).

1.1.2 Task 5 | Traitement uniforme

```
Quelle est la propriété caractéristique de l'orienté objet qui permet de
faire cela ?Qu'est-il spécifiquement nécessaire de faire dans le cas du C++
?  Quel est le type des éléments du tableau :  le tableau doit-il contenir
des objets ou des pointeurs vers ces objets ?  Pourquoi ?  Comparer à Java.
```

Réponse:

- Elle s'appelle polymorphism qui veut dire qu'on peut choisir le point de vue le plus approprié selon les besoins. Il faut déclarer ce type de méthode avec le mot clé `virtual` afin d'activer polymorphism **dynamique**, et `override` quand on veut reinplémenter la méthode.
- Le tableau doit contenir des pointers vers ces objets parce qu'un tableau ne peux contenir qu'un seul type d'objet.
- Il n'y a pas de notion de pointer en Java mais on a la notion de référance. D'où il n'y a pas de choix. On ne peut créer un tableau de référence.

1.1.3 Task 7 | Destruction et copie des objets

```
Parmi les classes précédemment écrites quelles sont celles qu'il faut modifier
afin qu'il n'y ait pas de fuite mémoire quand on détruit les objets ?  Modifiez
le code de manière à l'éviter.
```

Réponse:

- On vuet avoir la propriété de polymorphisme sur le méthode descruteur. Par conséquence, on ajoute le mot clé `virtual` devant le desctructeur de `class Base`.
- On modifie également the descructor of Film en ajoutant `delete[] chapiter_duration_table;`

```
La copie d'objet peut également poser problème quand ils ont des variables
d'instance qui sont des pointeurs.  Quel est le problème et quelles sont
les solutions ?
```

Réponse:

- Si on ne fait pas la modification, le copy operator copie juste la valeur de pointer. A la fin de opération, on a deux l'objets film dont leur chapiter_duration_table pointe au même tableau dans le mémoire. Par exemple:
  ```
  film film1 = film();
  film film2 = film1;
  ```
  on a finalement `film1.chapiter_duration_table = film.chapiter_duration_table`
- On doit reimplémenter le copy constructeur et = operator.

1.1.4 Task 8 | Créer des groupes

```
On rappelle aussi que la liste d'objets doit en fait être une liste de pointeurs
d'objets.  Pourquoi ?  Comparer à Java.
```

Réponse:

C'est déjà répondu dans la question de Task5.

1.1.5 Task 11 | client/serveur

Les commandes sont:

- GET <filename> : Search the `filename` object in the data base
- PLAY <filename> : Play the `filename` object on the side server.

Pour tester, 3 objets multimédia sont mis dans le dossier /cpp/media aussi le database de serveur. Les 3 objets sont

- imag1.jpg
- imag2.jpg
- film1.mp4

1.2 Partie Java Swing

1.2.1 Task 3:

Exemple d'utilisation:

- SEARCH : écrire le nom de l'objet Multimedia à rechercher dans le "JTextArea"
- PLAY : écrire le nom de l'objet Multimedia qu'on veut "jouer". Il sera lancé côté serveur.
- CLOSE : Fermer le client swing

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

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

Chapter 3

Class Index

3.1 Class List

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

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

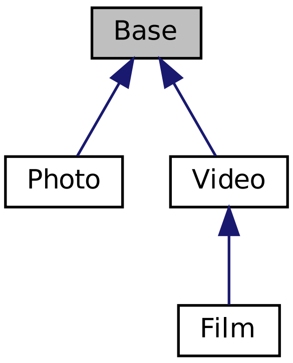
Chapter 5

Class Documentation

5.1 Base Class Reference

```
#include <base.h>
```

Inheritance diagram for Base:



Public Member Functions

- Base ()

  defaut constructeur
- Base (std::string _filename, std::string _file_path)

  Constructeur.
- virtual ∼Base ()
- std::string get_filename () const

  Getting the filename of object.
- std::string get_file_path () const

  Getting the file Path of object.

- void set_filename (std::string _filename)

  Set the filename of object.
- void set_file_path (std::string _file_path)

  Set the path of object.
- virtual void print (std::ostream &) const

  Print the detail of object.
- virtual void run () const =0

  "Play" the object

5.1.1 Constructor & Destructor Documentation

5.1.1.1 Base() [1/2]

```
Base::Base ( )
```

defaut constructeur

5.1.1.2 Base() [2/2]

```
Base::Base (
              std::string _filename,
              std::string _file_path )
```

Constructeur.

Parameters

_filename	The filename of object
_file_path	The path of oject

5.1.1.3 ∼Base()

```
Base::∼Base ( )  [virtual]
```

5.1.2 Member Function Documentation

5.1.2.1 get_file_path()

```
std::string Base::get_file_path ( ) const
```

Getting the file Path of object.

Returns

the filename of associated object

5.1.2.2 get_filename()

```
std::string Base::get_filename ( ) const
```

Getting the filename of object.

Returns

the filename of associated object

5.1.2.3 print()

```
void Base::print (
             std::ostream & out_stream ) const  [virtual]
```

Print the detail of object.

Parameters

the	output stream (For example `std::cout`)

Reimplemented in Video, Photo, and Film.

5.1.2.4 run()

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

"Play" the object

Implemented in Video, and Photo.

5.1.2.5 set_file_path()

```
void Base::set_file_path (
             std::string _file_path )
```

Set the path of object.

Parameters

_file_path	

5.1.2.6 set_filename()

```
void Base::set_filename (
             std::string _filename )
```

Set the filename of object.

Parameters

_filename	

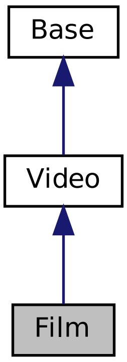
The documentation for this class was generated from the following files:

- cpp/base.h
- cpp/base.cpp

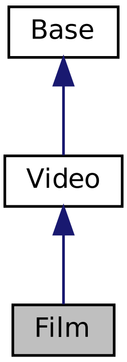
5.2 Film Class Reference

```
#include <film.h>
```

Inheritance diagram for Film:



Collaboration diagram for Film:



Public Member Functions

- Film ()

    Defaut constructor

- Film (int nb_chapiter, int _video_duration, std::string _filename, std::string _file_path)

    constructor

- ∼Film ()

    default destructor

- Film (const Film &from)

    copy constructor

- Film & operator= (const Film &from)

  redefine = operator
- void print (std::ostream &out_stream) const override

  print the detail of film
- void set_chapiter_table (int const ∗tab, int tab_len)

  set up the chapiter table
- int const ∗ get_chapiter_duration_table () const

  get the chapiter table
- int get_nb_chapiter () const

  get the length of chapiter table

5.2.1 Constructor & Destructor Documentation

5.2.1.1 Film() [1/3]

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

Defaut constructor

5.2.1.2 Film() [2/3]

```
Film::Film (
              int nb_chapiter,
              int _video_duration,
              std::string _filename,
              std::string _file_path )
```

constructor

Parameters

nb_chapiter	number of chapiter of video
_video_duration	the length of video
_filename	filename of video
_file_path	path of video

5.2.1.3 ∼Film()

```
Film::∼Film ( )
```

default destructor

5.2.1.4 Film() [3/3]

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

copy constructor

Parameters

from	the copy object target

Returns

a copy of from

5.2.2 Member Function Documentation

5.2.2.1 get_chapiter_duration_table()

```
int const * Film::get_chapiter_duration_table ( ) const
```

get the chapiter table

Returns

return the chapiter table

5.2.2.2 get_nb_chapiter()

```
int Film::get_nb_chapiter ( ) const
```

get the length of chapiter table

Returns

return the length of chapiter table

5.2.2.3 operator=()

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

redefine = operator

Parameters

from	the copy object target

Returns

a copy of from

5.2.2.4 print()

```
void Film::print (
             std::ostream & out_stream ) const  [override], [virtual]
```

print the detail of film

Parameters

out_stream	the outputstream (for example `std::cout`)

Reimplemented from Base.

5.2.2.5 set_chapiter_table()

```
void Film::set_chapiter_table (
             int const * tab,
             int tab_len )
```

set up the chapiter table

Parameters

tab	a list of int table
tab_len	length of list `tab`

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

- cpp/film.h
- cpp/film.cpp

5.3 Gestion Class Reference

```
#include <gestion.h>
```

Public Member Functions

- Gestion ()

  Construtor

- photoPtr create_photo (double _latitude, double _longitude, std::string _filename, std::string _file_path)

  creat the photo object and put in the tab_base
- videoPtr create_video (int _v_duration, std::string _filename, std::string _file_path)

  creat the video object and put in the tab_base
- filmPtr create_film (int nb_chapiter, int _video_duration, std::string _filename, std::string _file_path)

  creat the film object and put in the tab_base
- groupPtr create_group (std::string group_name)

  creat the group object and put in the tab_group
- void print (std::ostream &out_stream, std::string name) const

  print the detail of `name` object If no `name` object in the database, it print `name` not found! Otherwise, it print the detail information of object `name`
- void play (std::string filename) const

  run the multimedia object
- void delete_object (std::string filename)

  delete the `filename` object in the database
- void delete_group (std::string group_name)

  delete the `group_name` group in the database

5.3.1 Constructor & Destructor Documentation

5.3.1.1 Gestion()

```
Gestion::Gestion ( )
```

Construtor

5.3.2 Member Function Documentation

5.3.2.1 create_film()

```
filmPtr Gestion::create_film (
            int nb_chapiter,
            int _video_duration,
            std::string _filename,
            std::string _file_path )
```

creat the film object and put in the tab_base

Returns

a smart ponter of film that point to the film object

5.3.2.2 create_group()

```
groupPtr Gestion::create_group (
             std::string group_name )
```

creat the group object and put in the tab_group

Returns

a smart ponter of group that point to the group object

5.3.2.3 create_photo()

```
photoPtr Gestion::create_photo (
             double _latitude,
             double _longitude,
             std::string _filename,
             std::string _file_path )
```

creat the photo object and put in the tab_base

Returns

a smart ponter of photo that point to the photo object

5.3.2.4 create_video()

```
videoPtr Gestion::create_video (
             int _v_duration,
             std::string _filename,
             std::string _file_path )
```

creat the video object and put in the tab_base

Returns

a smart ponter of video that point to the video object

5.3.2.5 delete_group()

```
void Gestion::delete_group (
             std::string group_name )
```

delete the `group_name` group in the database

Parameters

group_name	The group_name of target group

5.3.2.6 delete_object()

```
void Gestion::delete_object (
             std::string filename )
```

delete the `filename` object in the database

Parameters

filename	the filename of target object.

5.3.2.7 play()

```
void Gestion::play (
             std::string filename ) const
```

run the multimedia object

Parameters

filemanem	the filename of mutimedia object

5.3.2.8 print()

```
void Gestion::print (
             std::ostream & out_stream,
             std::string name ) const
```

print the detail of `name` object If no `name` object in the database, it print `name` not found! Otherwise, it print the detail information of object `name`

Parameters

out_stream	example `std::out`
name	look up the "name" object in the database

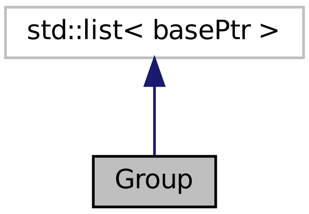
The documentation for this class was generated from the following files:

- cpp/gestion.h
- cpp/gestion.cpp

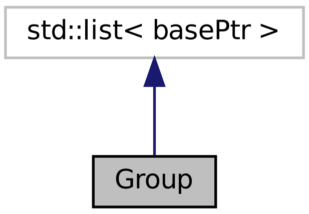
5.4 Group Class Reference

```
#include <group.h>
```

Inheritance diagram for Group:



Collaboration diagram for Group:



Public Member Functions

- Group (std::string name)

  Defaut Group constructoer.
- std::string get_group_name () const

  get the name of group
- void print (std::ostream &out_stream) const

  print the detail of group

5.4.1 Constructor & Destructor Documentation

5.4.1.1 Group()

```
Group::Group (
             std::string name )
```

Defaut Group constructoer.

Parameters

name	name of the group

5.4.2 Member Function Documentation

5.4.2.1 get_group_name()

```
std::string Group::get_group_name ( ) const
```

get the name of group

Returns

the name of group

5.4.2.2 print()

```
void Group::print (
             std::ostream & out_stream ) const
```

print the detail of group

Parameters

out_stream	the print the information into outstream for exampele (`std::cout`)

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

- cpp/group.h
- cpp/group.cpp

5.5 InputBuffer Struct Reference

Public Member Functions

- InputBuffer (size_t size)
- ∼InputBuffer ()

Public Attributes

- char ∗ buffer
- char ∗ begin
- char ∗ end
- SOCKSIZE remaining

5.5.1 Constructor & Destructor Documentation

5.5.1.1 InputBuffer()

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

5.5.1.2 ∼InputBuffer()

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

5.5.2 Member Data Documentation

5.5.2.1 begin

```
char∗ InputBuffer::begin
```

5.5.2.2 buffer

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

5.5.2.3 end

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

5.5.2.4 remaining

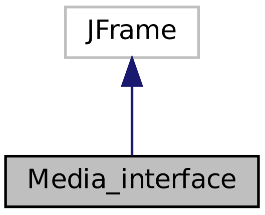
```
SOCKSIZE InputBuffer::remaining
```

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

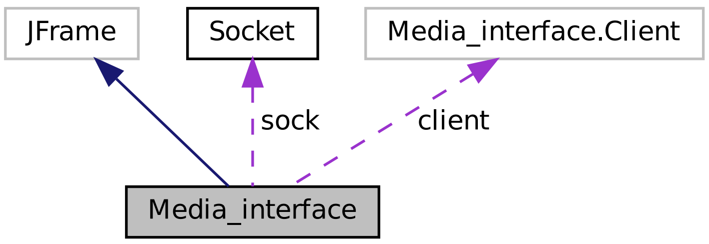
- cpp/ccsocket.cpp

5.6 Media_interface Class Reference

Inheritance diagram for Media_interface:



Collaboration diagram for Media_interface:



Classes

- class **Client**

  This class impelement the connection TCP to the server.
- class **CloseListener**

  This class impelement the CLOSE action for "close" button.
- class **GETaction**

  This class impelement the GET action for "Search" button.
- class **PLAYaction**

  This class impelement the PLAY action for "Play" button.

Public Member Functions

- Media_interface (String argv[ ])

  The constructor of Media_interface.

Static Public Member Functions

- static void main (String argv[ ])

5.6.1 Constructor & Destructor Documentation

5.6.1.1 Media_interface()

```
public Media_interface.Media_interface (
             String argv[ ] )  [inline]
```

The constructor of Media_interface.

Parameters

argv	host port

5.6.2 Member Function Documentation

5.6.2.1 main()

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

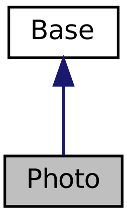
The documentation for this class was generated from the following file:

- swing/Media_interface.java

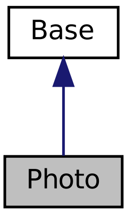
5.7 Photo Class Reference

```
#include <photo.h>
```

Inheritance diagram for Photo:



Collaboration diagram for Photo:



Public Member Functions

- Photo ()

  Defaut constructor.
- Photo (double _latitude, double _longitude, std::string _filename, std::string _file_path)

  constructor
- ∼Photo ()
- double get_latitude () const

  get the latitude of photo
- double get_longitude () const

  get the longitude of photo
- void set_latitude (double _latitude)

  set the latitude of photo
- void set_longitude (double _longitude)

  set the longitude of photo

- void print (std::ostream &out_stream) const override

  print the detail of photo
- void run () const override

  Play the photo.

5.7.1 Constructor & Destructor Documentation

5.7.1.1 Photo() [1/2]

```
Photo::Photo ( )
```

Defaut constructor.

5.7.1.2 Photo() [2/2]

```
Photo::Photo (
            double _latitude,
            double _longitude,
            std::string _filename,
            std::string _file_path )
```

constructor

Parameters

* *	

5.7.1.3 ∼Photo()

```
Photo::∼Photo ( )
```

5.7.2 Member Function Documentation

5.7.2.1 get_latitude()

```
double Photo::get_latitude ( ) const
```

get the latitude of photo

Returns

the latitude of photo

5.7.2.2 get_longitude()

```
double Photo::get_longitude ( ) const
```

get the longitude of photo

Returns

the longitude of photo

5.7.2.3 print()

```
void Photo::print (
             std::ostream & out_stream ) const  [override], [virtual]
```

print the detail of photo

Parameters

out_stream	the output stream (For example `std::out`)

Reimplemented from Base.

5.7.2.4 run()

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

Play the photo.

Implements Base.

5.7.2.5 set_latitude()

```
void Photo::set_latitude (
             double _latitude )
```

set the latitude of photo

Parameters

_latitude	the latitude value

5.7.2.6 set_longitude()

```
void Photo::set_longitude (
             double _longitude )
```

set the longitude of photo

Parameters

_longitude	the longitude value

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

- cpp/photo.h
- cpp/photo.cpp

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 or 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 fand stores them in *buf*. By default, this function blocks the caller until thre is availbale 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 occured.

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]
```

Send sdata 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 occured.

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]
```

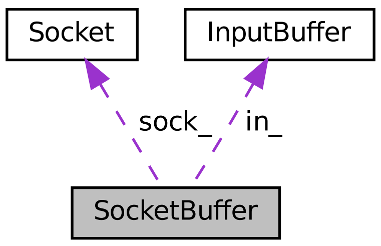
The documentation for this class was generated from the following files:

- 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 ouputSize=8192)
- SocketBuffer (Socket &, size_t inputSize=8192, size_t ouputSize=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 retreive the entire record. Beware messages should thus not contain these charecters.

```
 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 occured
- 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 occurences 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]
```

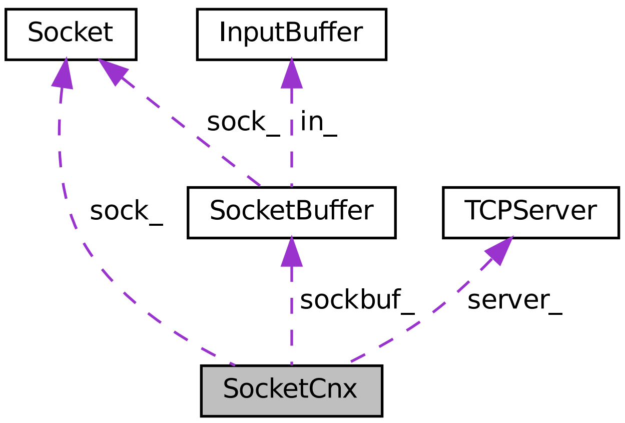
The documentation for this class was generated from the following files:

- 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]
```

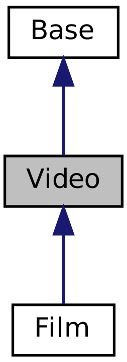
The documentation for this class was generated from the following files:

- cpp/tcpserver.h
- cpp/tcpserver.cpp

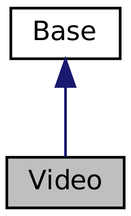
5.13 Video Class Reference

```
#include <video.h>
```

Inheritance diagram for Video:



Collaboration diagram for Video:



Public Member Functions

- Video ()
  defaut constructor
- Video (int _video_duration, std::string _filename, std::string _file_path)
  constructor
- ∼Video ()
- int get_video_duration () const
  get the length of video
- void set_video_duration (int _video_duration)
  set the length of video
- void print (std::ostream &out_stream) const override
  print the detail of video
- void run () const override
  play the video

5.13.1 Constructor & Destructor Documentation

5.13.1.1 Video() `[1/2]`

```
Video::Video ( )
```

defaut constructor

5.13.1.2 Video() `[2/2]`

```
Video::Video (
          int _video_duration,
          std::string _filename,
          std::string _file_path )
```

constructor

Parameters

_video_duration	The length of video
_filename	The filename of video
_file_path	The path of video

5.13.1.3 ∼Video()

```
Video::∼Video ( )
```

Parameters

descructor	

5.13.2 Member Function Documentation

5.13.2.1 get_video_duration()

```
int Video::get_video_duration ( ) const
```

get the length of video

Returns

the lenght of video

5.13.2.2 print()

```
void Video::print (
            std::ostream & out_stream ) const  [override], [virtual]
```

print the detail of video

Parameters

out_stream	The output stream (for example `std::out`)

Reimplemented from Base.

5.13.2.3 run()

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

play the video

Implements Base.

5.13.2.4 set_video_duration()

```
void Video::set_video_duration (
             int _video_duration )
```

set the length of video

Parameters

_video_duration	The length of video

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

- cpp/video.h
- cpp/video.cpp

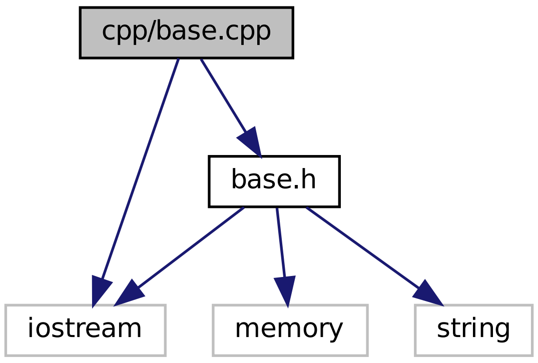
Chapter 6

File Documentation

6.1 cpp/base.cpp File Reference

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

Include dependency graph for base.cpp:

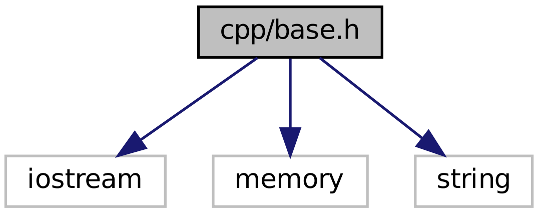


6.2 cpp/base.h File Reference

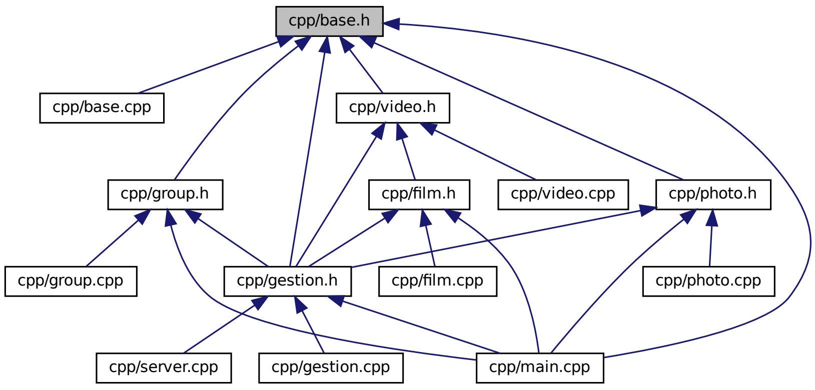
class Base gives the basic structure of mutimedia objects.

```
#include <iostream>
#include <memory>
```

`#include <string>`
Include dependency graph for base.h:



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



Classes

- class Base

Typedefs

- using basePtr = std::shared_ptr< Base >

6.2.1 Detailed Description

class Base gives the basic structure of mutimedia objects.

Author

Cheng-Yen Wu

Date

2022

6.2.2 Typedef Documentation

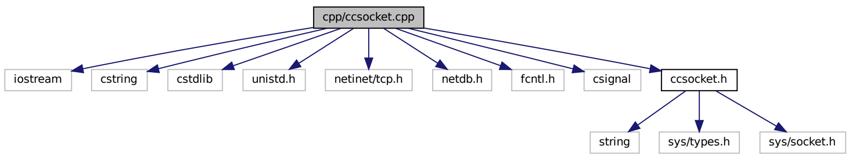
6.2.2.1 basePtr

```
using basePtr = std::shared_ptr<Base>
```

6.3 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:



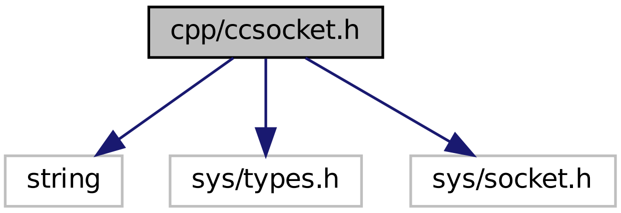
Classes

- struct InputBuffer

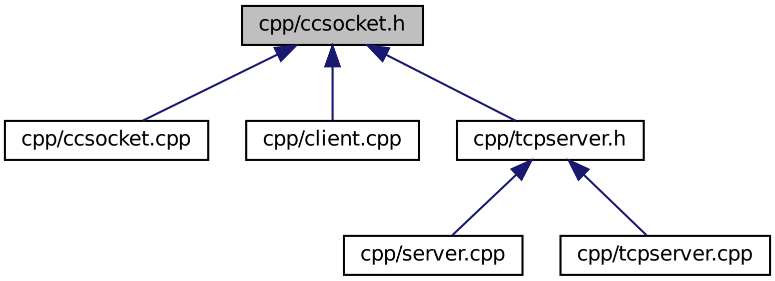
6.4 cpp/ccsocket.h File Reference

```
#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)

6.4.1 Macro Definition Documentation

6.4.1.1 INVALID_SOCKET

```
#define INVALID_SOCKET -1
```

6.4.1.2 NO_SIGPIPE_

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

6.4.1.3 SOCKADDR

```
#define SOCKADDR struct sockaddr
```

6.4.1.4 SOCKADDR_IN

```
#define SOCKADDR_IN struct sockaddr_in
```

6.4.1.5 SOCKDATA

```
#define SOCKDATA void
```

6.4.1.6 SOCKET

```
#define SOCKET int
```

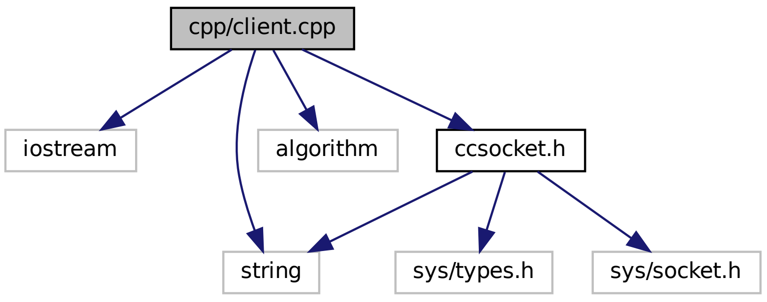
6.4.1.7 SOCKSIZE

```
#define SOCKSIZE ssize_t
```

6.5 cpp/client.cpp File Reference

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

Include dependency graph for client.cpp:



Functions

- int main ()

6.5.1 Function Documentation

6.5.1.1 main()

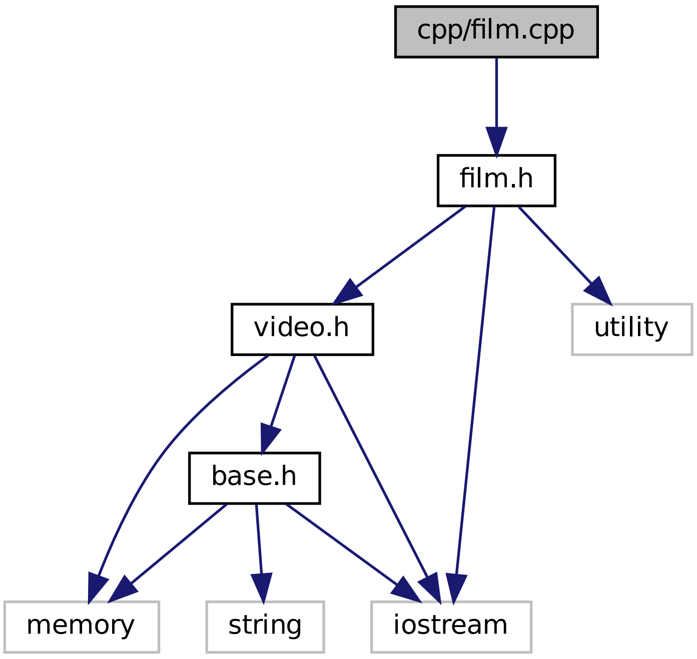
```
int main ( )
```

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.

6.6 cpp/film.cpp File Reference

```
#include "film.h"
```

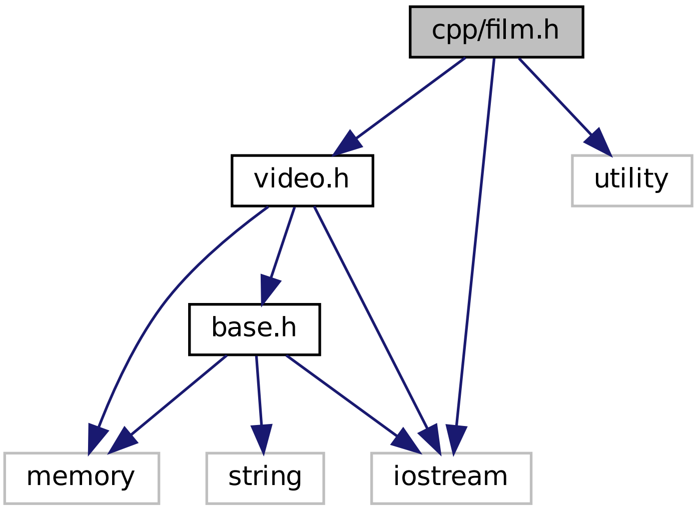
Include dependency graph for film.cpp:



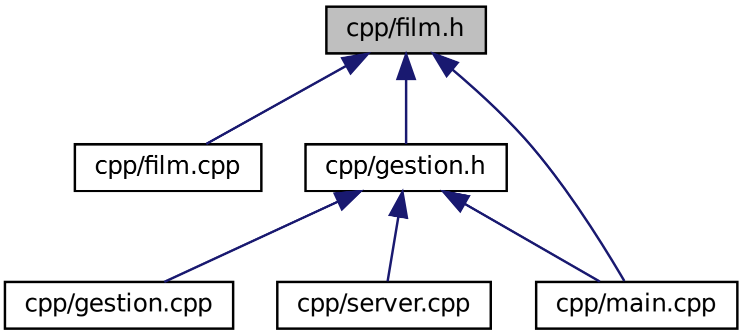
6.7 cpp/film.h File Reference

```
#include "video.h"
#include <iostream>
#include <utility>
```

Include dependency graph for film.h:



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



Classes

- class Film

Typedefs

- using filmPtr = std::shared_ptr< Film >

6.7.1 Typedef Documentation

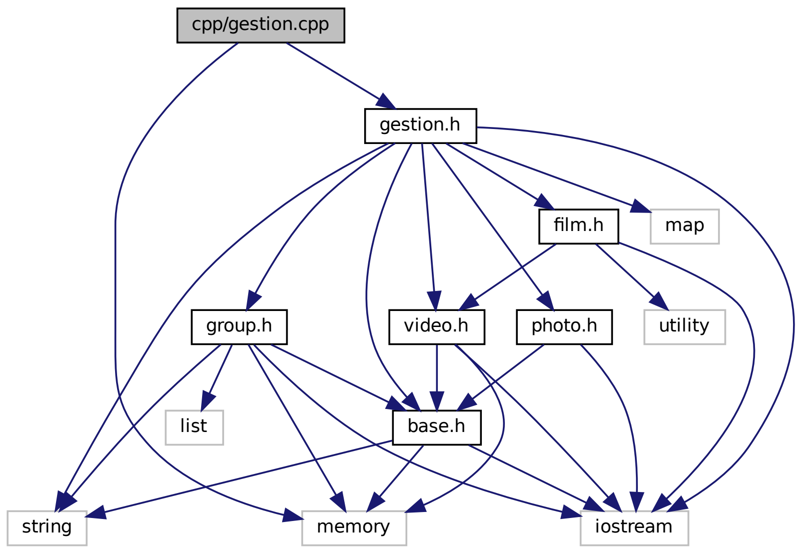
6.7.1.1 filmPtr

```
using filmPtr = std::shared_ptr<Film>
```

6.8 cpp/gestion.cpp File Reference

```
#include "gestion.h"
#include <memory>
```

Include dependency graph for gestion.cpp:



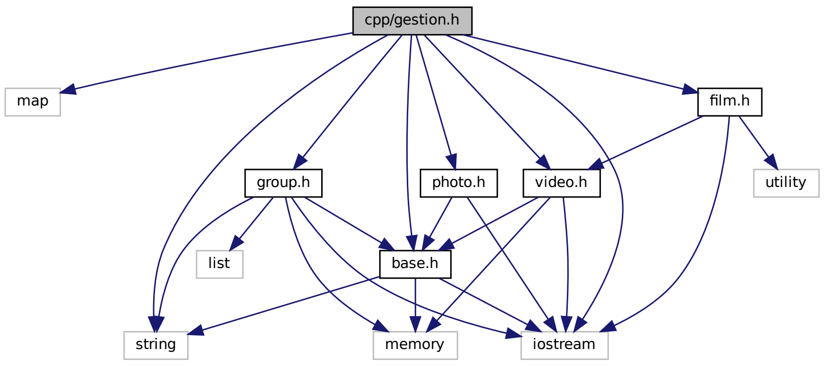
6.9 cpp/gestion.h File Reference

class Gestion provide a detail structure of object photo

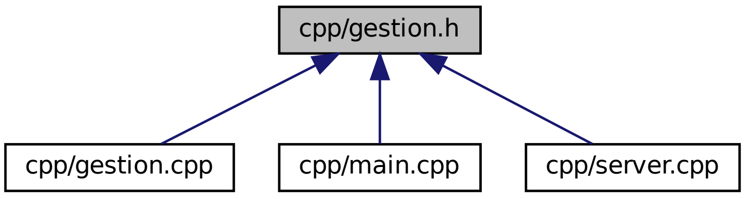
```
#include <map>
#include <string>
#include "group.h"
#include "base.h"
```

```
#include "film.h"
#include "photo.h"
#include "video.h"
#include <iostream>
```

Include dependency graph for gestion.h:



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



Classes

- class Gestion

6.9.1 Detailed Description

class Gestion provide a detail structure of object photo

Author

Cheng-Yen Wu

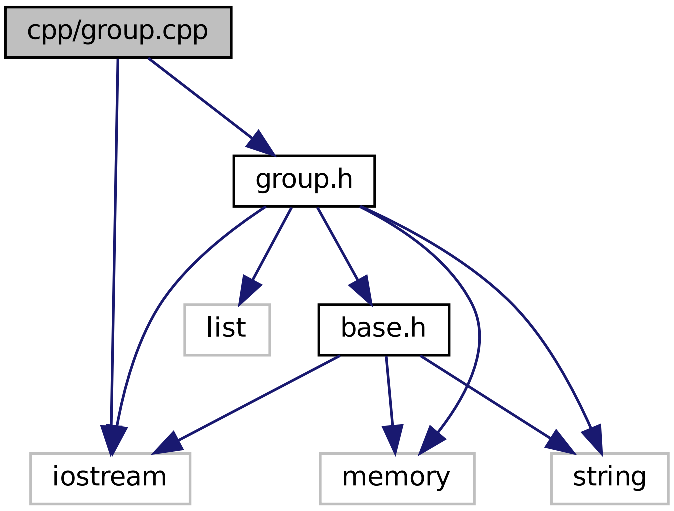
Date

2022

6.10 cpp/group.cpp File Reference

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

Include dependency graph for group.cpp:

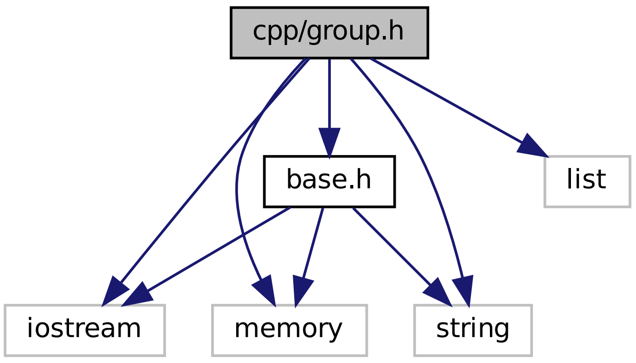


6.11 cpp/group.h File Reference

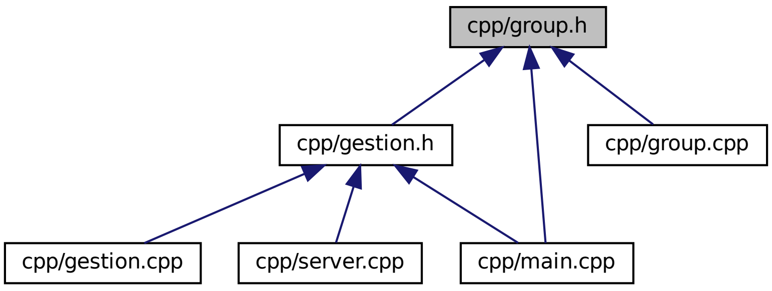
class Group provide a collection struture of multimedia object

```
#include "base.h"
#include <list>
#include <string>
#include <iostream>
#include <memory>
```

Include dependency graph for group.h:



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



Classes

- class Group

Typedefs

- using groupPtr = std::shared_ptr< Group >

6.11.1 Detailed Description

class Group provide a collection struture of multimedia object

Author

Cheng-Yen Wu

Date

2022

6.11.2 Typedef Documentation

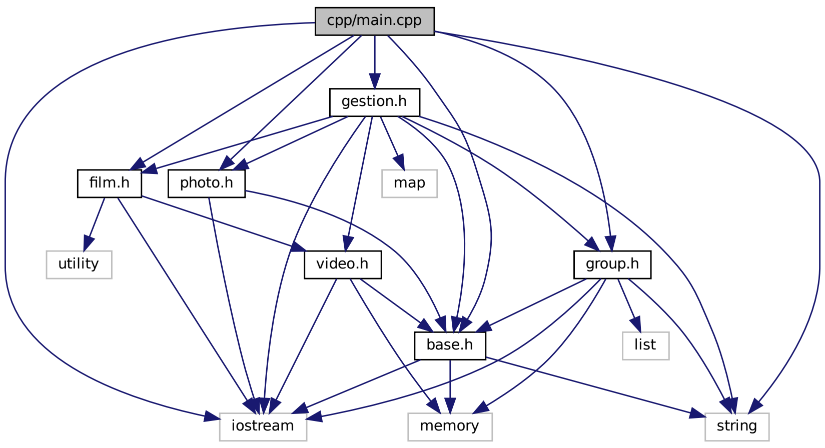
6.11.2.1 groupPtr

```
using groupPtr = std::shared_ptr<Group>
```

6.12 cpp/main.cpp File Reference

```
#include <iostream>
#include <string>
#include "base.h"
#include "photo.h"
#include "film.h"
#include "group.h"
#include "gestion.h"
```

Include dependency graph for main.cpp:



Functions

- void test4 ()

    Simple test for task 4.
- void test5 ()

    Simple test for task 5.
- void test6 ()

    simple test for task 6
- void test7 ()

    simple test for task 7
- void test9 ()

    simple test for task 9
- void test10 ()

    simple test for task 10
- int main (int argc, const char ∗argv[ ])

Variables

- string media_path = "/home/cheng-yen/Documents/X/4A/inf224/tp/media/"

6.12.1 Detailed Description

Date

2022

Author

Cheng-Yen Wu

6.12.2 Function Documentation

6.12.2.1 main()

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

6.12.2.2 test10()

```
void test10 ( )
```

simple test for task 10

6.12.2.3 test4()

```
void test4 ( )
```

Simple test for task 4.

6.12.2.4 test5()

```
void test5 ( )
```

Simple test for task 5.

6.12.2.5 test6()

```
void test6 ( )
```

simple test for task 6

6.12.2.6 test7()

```
void test7 ( )
```

simple test for task 7

6.12.2.7 test9()

```
void test9 ( )
```

simple test for task 9

6.12.3 Variable Documentation

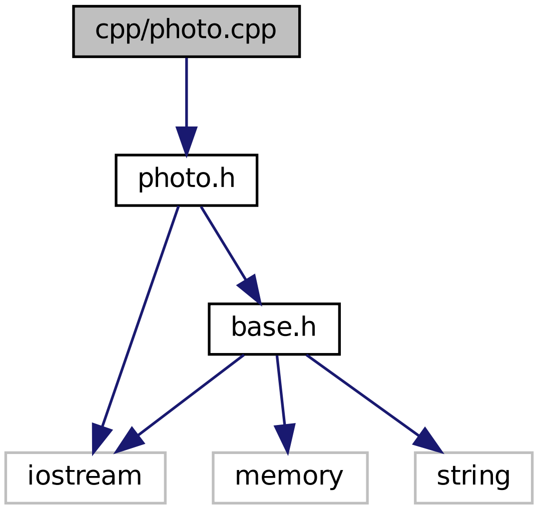
6.12.3.1 media_path

```
string media_path = "/home/cheng-yen/Documents/X/4A/inf224/tp/media/"
```

6.13 cpp/photo.cpp File Reference

```
#include "photo.h"
```

Include dependency graph for photo.cpp:

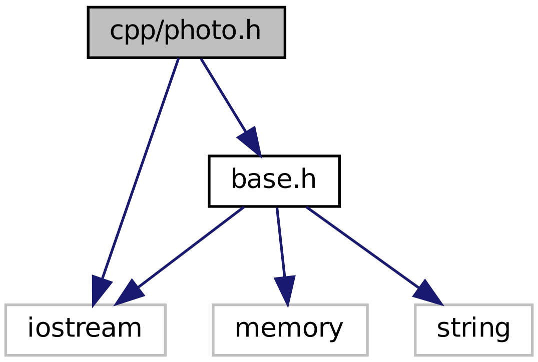


6.14 cpp/photo.h File Reference

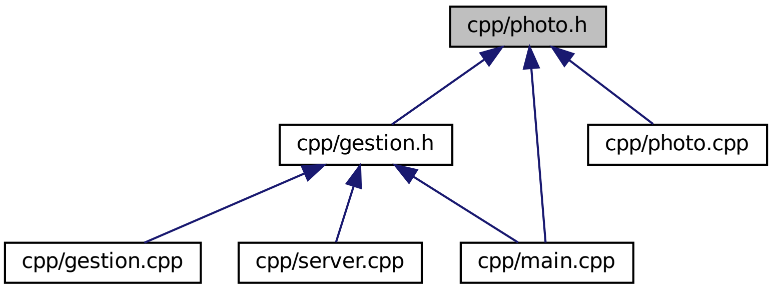
class Photo provide a detail structure of object photo

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

Include dependency graph for photo.h:



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



Classes

- class Photo

Typedefs

- using photoPtr = std::shared_ptr< Photo >

6.14.1 Detailed Description

class Photo provide a detail structure of object photo

Author

Cheng-Yen Wu

Date

2022

6.14.2 Typedef Documentation

6.14.2.1 photoPtr

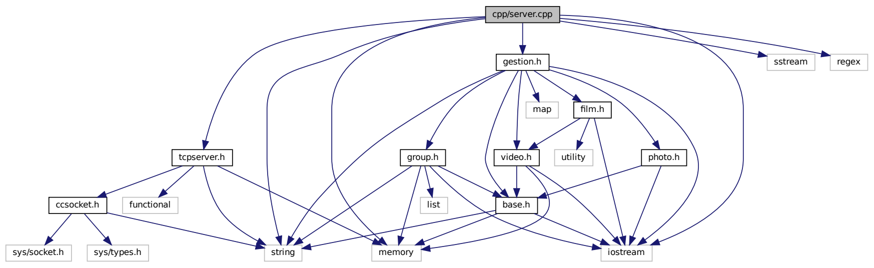
```
using photoPtr = std::shared_ptr<Photo>
```

6.15 cpp/README.md File Reference

6.16 cpp/server.cpp File Reference

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

Include dependency graph for server.cpp:



Functions

- int main (int argc, char ∗argv[ ])

Variables

- const int PORT = 3331
- const std::string media_path ="./media/"
- Gestion db = Gestion()
- auto pho1 = db.create_photo(11,11, "imag1.jpg", media_path+"imag1.jpg")
- auto pho2 = db.create_photo(22,22, "imag2.jpg", media_path+"imag2.jpg")
- auto film1 = db.create_film(1, 3, "film1.mp4", media_path+"film1.mp4")
- auto gr1 = db.create_group("gr1")

6.16.1 Function Documentation

6.16.1.1 main()

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

6.16.2 Variable Documentation

6.16.2.1 db

```
Gestion db = Gestion()
```

6.16.2.2 film1

```
auto film1 = db.create_film(1, 3, "film1.mp4", media_path+"film1.mp4")
```

6.16.2.3 gr1

```
auto gr1 = db.create_group("gr1")
```

6.16.2.4 media_path

```
const std::string media_path ="./media/"
```

6.16.2.5 pho1

```
auto pho1 = db.create_photo(11,11, "imag1.jpg", media_path+"imag1.jpg")
```

6.16.2.6 pho2

```
auto pho2 = db.create_photo(22,22, "imag2.jpg", media_path+"imag2.jpg")
```

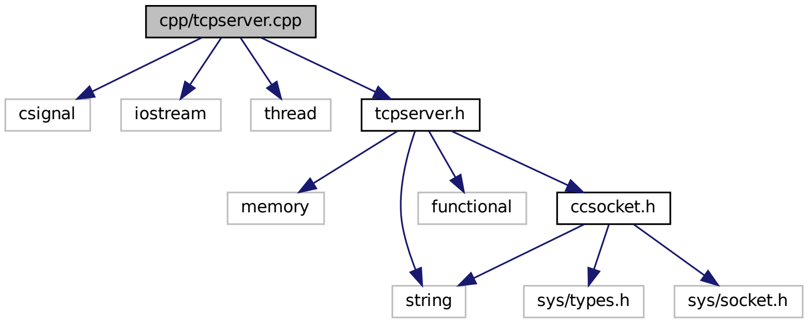
6.16.2.7 PORT

```
const int PORT = 3331
```

6.17 cpp/tcpserver.cpp File Reference

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

Include dependency graph for tcpserver.cpp:



Classes

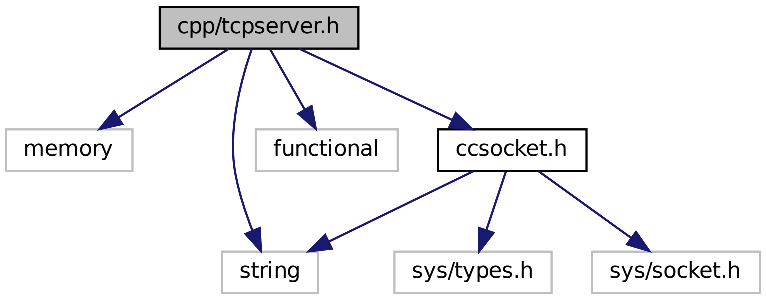
- class SocketCnx

  Connection with a given client. Each SocketCnx uses a different thread.

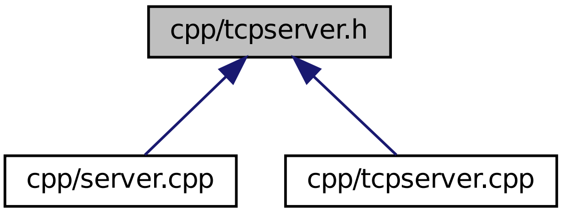
6.18 cpp/tcpserver.h File Reference

```
#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:



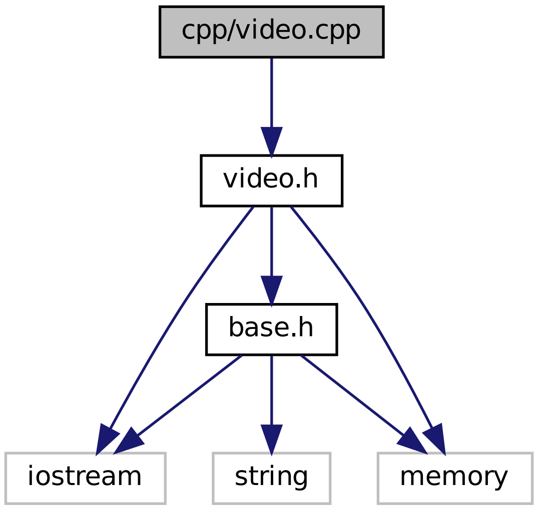
Classes

- class TCPServer

6.19 cpp/video.cpp File Reference

```
#include "video.h"
```

Include dependency graph for video.cpp:

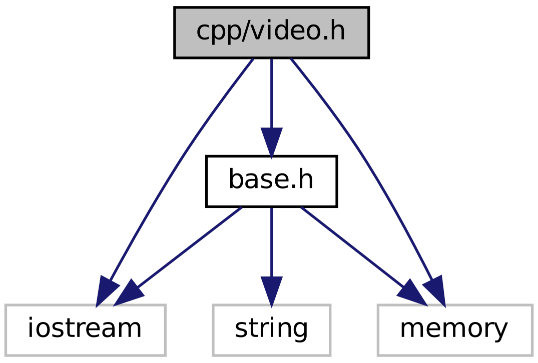


6.20 cpp/video.h File Reference

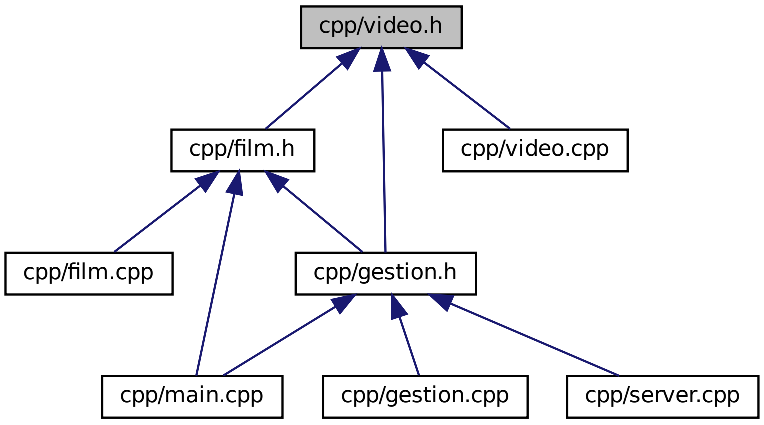
class Video provide a detail structure of object video.

```
#include "base.h"
#include <memory>
#include <iostream>
```

Include dependency graph for video.h:



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



Classes

- class Video

Typedefs

- using videoPtr = std::shared_ptr< Video >

6.20.1 Detailed Description

class Video provide a detail structure of object video.

Author

Cheng-Yen Wu

Date

2022

6.20.2 Typedef Documentation

6.20.2.1 videoPtr

```
using videoPtr = std::shared_ptr<Video>
```

6.21 swing/Media_interface.java File Reference

Classes

- class Media_interface
- class **Media_interface.GETaction**

  This class impelement the GET action for "Search" button.
- class **Media_interface.PLAYaction**

  This class impelement the PLAY action for "Play" button.
- class **Media_interface.CloseListener**

  This class impelement the CLOSE action for "close" button.
- class **Media_interface.Client**

  This class impelement the connection TCP to the server.

6.21.1 Detailed Description

This interface is writen in java swing. It listens to 127.0.0.1:3331 by defaut and try to connect to the server implemented in ../cpp/server.cpp

Index