yasio Documentation

yasio is a multi-platform support c++11 library with focus on asio (asynchronous socket I/O) for any client application.

- Cross-platform:
  - Compiler:
    - Visual Studio 2013+
    - GCC4.7+
    - xcode9+
    - Other C++11,14,17 Compilers
  - Architecture: x86, x64, ARM and etc.
  - OS: Windows, macOS, Linux, FreeBSD, iOS, Android And etc.
- Open source location: 🔠@🔠

Quick Start

This demo simply send http request to `tool.chinaz.com` and print resposne data.

C++

```cpp
#include "yasio/yasio.hpp"
#include "yasio/obstream.hpp"
using namespace yasio;
using namespace yasio::inet;
int main()
{
    io_service service({"tool.chinaz.com", 80});
    service.set_option(YOPT_S_DEFERRED_EVENT, 0); // dispatch network event on network thread
    service.start([&](event_ptr&& ev) {
        switch (ev->kind())
        {
        case YEK_ON_PACKET: {
            auto packet = std::move(ev->packet());
            fwrite(packet.data(), packet.size(), 1, stdout);
            fflush(stdout);
            break;
        }
        case YEK_ON_OPEN:
            if (ev->status() == 0)
            {
                auto transport = ev->transport();
                if (ev->cindex() == 0)
                {
                    obstream obs;
                    obs.write_bytes("GET /index.htm HTTP/1.1\r\n");

                    obs.write_bytes("Host: tool.chinaz.com\r\n");

                    obs.write_bytes("User-Agent: Mozilla/5.0 (Windows NT 10.0; "
                                    "WOW64) AppleWebKit/537.36 (KHTML, like Gecko) "
                                    "Chrome/87.0.4820.88 Safari/537.36\r\n");
                    obs.write_bytes("Accept: */*;q=0.8\r\n");
                    obs.write_bytes("Connection: Close\r\n\r\n");

                    service.write(transport, std::move(obs.buffer()));
                }
            }
            break;
        case YEK_ON_CLOSE:
            printf("The connection is lost.\n");
            break;
        }
    });
    // open channel 0 as tcp client
    service.open(0, YCK_TCP_CLIENT);
    getchar();
}
```

Lua

```lua
local ip138 = "tool.chinaz.com"
local service = yasio.io_service.new({host=ip138, port=80})
local respdata = ""
```

```
service:start(function(ev)
        local k = ev.kind()
        if (k == yasio.YEK_ON_PACKET) then
            respdata = respdata .. ev:packet():to_string()
        elseif k == yasio.YEK_ON_OPEN then
            if ev:status() == 0 then -- connect succeed
                local transport = ev:transport()
                local obs = yasio.obstream.new()
                obs:write_bytes("GET / HTTP/1.1\r\n")

                obs:write_bytes("Host: " .. ip138 .. "\r\n")

                obs:write_bytes("User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/
537.36 (KHTML, like Gecko) Chrome/79.0.3945.117 Safari/537.36\r\n")
                obs:write_bytes("Accept: */*;q=0.8\r\n")
                obs:write_bytes("Connection: Close\r\n\r\n")

                service:write(transport, obs)
            end
        elseif k == yasio.YEK_ON_CLOSE then
            print("request finish, respdata: " ..  respdata)
        end
    end)
-- Open channel 0 as tcp client and start non-blocking tcp 3 times handshake
service:open(0, yasio.YCK_TCP_CLIENT)

-- Call this function at thread which focus on the network event.
function gDispatchNetworkEvent(...)
    service:dispatch(128) -- dispatch max events is 128 per frame
end

_G.yservice = service -- Store service to global table as a singleton instance
```

The tests & examples

- tests:
  - echo_server: TCP/UDP/KCP echo server
  - echo_client: TCP/UDP/KCP echo client
  - ssltest: SSL client test, Get github.com home page
  - tcptest: TCP test
  - speedtest: TCP,UDP,KCP local transfer
  - mcast: multi-cast test program
- examples:
  - ftp_server: A simple ftp server only support file download which is based on yasio click to visit.

- lua: lua test contains http request TCP unpack test code.
- xlua: Unity3D xlua Integration Demo.
- DemoUE4: Unreal Engine 4 Integration Demo.

Build tests & examples

- Ensure install compiler which support C++11, such as `msvc`, `gcc`, `clang`
- Ensure `git`, `cmake` installed
- Execute follow commands:

```
git clone https://github.com/yasio/yasio
cd yasio
git submodule update --init --recursive
cd build
# for xcode should be: cmake .. -GXcode
cmake ..
cmake --build . --config Debug
```

io_service Class

Provides the functionality of `tcp, udp, kcp and ssl-client` communication with noblocking-io model.

Syntax

```
namespace yasio { namespace inet { class io_service; } }
```

Members

Public Constructors

Name	Description
io_service::io_service	Constructs a `io_service` object.

Public Methods

Name	Description
io_service::start	Start the network service thread.
io_service::stop	Stop the network service thread.
io_service::open	Open channel.
io_service::close	Close transport.
io_service::is_open	Tests whether channel or transport is open.
io_service::dispatch	Dispatch the network io events.
io_service::write	Sends data asynchronous.
io_service::write_to	Sends data to specific remote asynchronous.

Name	Description
io_service::schedule	Save the stream binary data to file.
io_service::init_globals	Init global data with print function callback.
io_service::cleanup_globals	Cleanup the global print function callback.
io_service::channel_at	Retrieves the channel by index.
io_service::set_option	Set options.

Remarks

By default, the transport use object_pool.

Requirements

Header: yasio.hpp

io_service::io_service

Constructs a `io_service` object.

```
io_service::io_service();

io_service::io_service(int channel_count);

io_service::io_service(const io_hostent& channel_ep);

io_service::io_service(const io_hostent* channel_eps, int channel_count);
```

Parameters

channel_count
The channel count.

channel_ep
The channel endpoint.

channel_eps
The first pointer of channel endpoints.

Example

```
#include "yasio/yasio.hpp"
int main() {
    using namespace yasio;
    using namespace yasio::inet;
    io_service s1; // s1 only support 1 channel
    io_service s2(5); // s2 support 5 channels concurrency
    io_service s3(io_hostent{"github.com", 443}); // s3 support 1 channel
    io_hostent hosts[] = {
        {"192.168.1.66", 20336},
        {"192.168.1.88", 20337},
    };
    io_service s4(hosts, YASIO_ARRAYSIZE(hosts)); // s4 support 2 channels concurrency
    return 0;
}
```

io_service::start

Start the network service thread.

```
void start(io_event_cb_t cb);
```

Parameters

cb
The callback to receive network io events.

Example

```
#include "yasio/yasio.hpp"
int main() {
    using namespace yasio;
    using namespace yasio::inet;
    auto service = yasio_shared_service(io_hostent{host="ip138.com", port=80});
    service->start([](event_ptr&& ev) {
    auto kind = ev->kind();
    if (kind == YEK_CONNECT_RESPONSE)
    {
        if (ev->status() == 0)
        printf("[%d] connect succeed.\n", ev->cindex());
        else
        printf("[%d] connect failed!\n", ev->cindex());
    }
    });
```

```
    return 0;
}
```

io_service::stop

Stop network service thread.

```
void stop()
```

Remarks

If the network service thread running, this function will post exit signal and wait it exit properly.

Example

TODO:

io_service::open

Open a channel.

```
void open(size_t cindex, int kind);
```

Parameters

cindex
The index of channel.

kind
The kind of channel.

Remarks

For tcp, will start the non-blocking 3 times handshake to establish tcp connection.

The *cindex* value must be less than max channels supported by this io_service.

The *kind* must be follow values

- `YCK_TCP_CLIENT`
- `YCK_TCP_SERVER`
- `YCK_UDP_CLIENT`

- `YCK_UDP_SERVER`
- `YCK_KCP_CLIENT`
- `YCK_KCP_SERVER`
- `YCK_SSL_CLIENT`

Example

TODO:

io_service::close

Close the channel or transport.

```
void close(transport_handle_t transport);

void close(int cindex);
```

Parameters

transport
The transport to be close.

cindex
The channel index to be close.

Remarks

For tcp, will trigger 4 times handsake to terminate the connection.

Example

TODO:

io_service::is_open

Tests whether the transport or channel is open.

```
bool is_open(transport_handle_t transport) const;

bool is_open(int cindex) const;
```

Parameters

transport
The transport to be tests.

cindex
The index of channel to be tests.

Example

TODO:

io_service::dispatch

Consume network events queue and dispatch them.

```
void dispatch(int max_count);
```

Parameters

max_count
The max count allow to dispatch at this time.

Remarks

Usually, this function should call at logic thread, such as cocos2d-x render thread or other game engine main thread.

It's useful to update game ui safety.

Example

```
yasio_shared_service()->dispatch(128);
```

io_service::write

Sends data asynchronous.

```
int write(
    transport_handle_t thandle,
    std::vector<char> buffer,
    io_completion_cb_t completion_handler = nullptr
);
```

Parameters

thandle
The transport handle to send.

buffer
The send buffer.

completion_handler
The completion handler for send operation.

Return Value

A number of bytes to sends, error occured when < 0.

Remarks

The *completion_handler* not support KCP.

The empty buffer will be ignored and not trigger completion_handler.

Example

TODO:

io_service::write_to

Sends data asynchronous.

```
int write_to(
    transport_handle_t thandle,
    std::vector<char> buffer,
    const ip::endpoint& to,
    io_completion_cb_t completion_handler = nullptr
);
```

Parameters

thandle
The transport handle to send.

buffer
The send buffer.

to
The remote endpoint for send operation.

completion_handler
The completion handler for send operation.

Return Value

A number of bytes to be send, error occured when < 0.

Remarks

This function only works for *DGRAM* transport `udp,kcp`

The *completion_handler* not support KCP.

The empty buffer will be ignored and not trigger completion_handler.

Example

TODO:

io_service::schedule

Schedule a timer which will dispatch on the network service thread.

```
highp_timer_ptr schedule(
    const std::chrono::microseconds& duration,
    timer_cb_t cb
);
```

Parameters

duration
The timer expire duration.

cb
The callback to execute when the timer is expired.

Return Value

The shared_ptr of the high resolution timer.

Example

```
// Register a once timer, timeout is 3 seconds.
yasio_shared_service()->schedule(std::chrono::seconds(3), []()->bool{
  printf("time called!\n");
  return true;
});

// Register a loop timer, interval is 5 seconds.
auto loopTimer = yasio_shared_service()->schedule(std::chrono::seconds(5), []()->bool{
  printf("time called!\n");
  return false;
});
```

io_service::init_globals

Explicit init global data with print function callback.

```
static void init_globals(print_fn2_t print_fn);
```

Parameters

print_fn
The custom print function to print network service log.

Remarks

This function is optional, it's useful to redirect network service log to your custom log system, such as ue4,u3d, see the example.

Example

```
// yasio_uelua.cpp
// compile with: /EHsc
#include "yasio_uelua.h"
#include "yasio/platform/yasio_ue4.hpp"
#include "lua.hpp"
#if defined(NS_SLUA)
using namespace NS_SLUA;
#endif
#include "yasio/bindings/lyasio.cpp"

DECLARE_LOG_CATEGORY_EXTERN(yasio_ue4, Log, All);
DEFINE_LOG_CATEGORY(yasio_ue4);

void yasio_uelua_init(void* L)
{
```

```
  auto Ls            = (lua_State*)L;
  print_fn2_t log_cb = [](int level, const char* msg) {
    FString text(msg);
    const TCHAR* tstr = *text;
    UE_LOG(yasio_ue4, Log, L"%s", tstr);
  };
  io_service::init_globals(log_cb);

  luaregister_yasio(Ls);
}
void yasio_uelua_cleanup()
{
  io_service::cleanup_globals();
}
```

io_service::cleanup_globals

Clear custom print function object.

```
static void cleanup_globals();
```

Remarks

You should call this function before unload a module(.dll,.so) which contains custom print function object.

io_service::channel_at

Retrieves channel by index.

```
io_channel* channel_at(size_t cindex) const;
```

Parameters

cindex
The index of channel.

Return value

The channel pointer, will be `nullptr` if the index out-of-range.

io_service::set_option

Set current io_service option.

```
void set_option(int opt, ...);
```

Parameters

opt
The opt value, see YOPT_X_XXX.

Example

```
#include "yasio/yasio.hpp"

int main(){
    using namespace yasio;
    using namespace yasio::inet;
    io_hostent hosts[] = {
    {"192.168.1.66", 20336},
    {"192.168.1.88", 20337},
    };
    auto service = std::make_shared<io_service>(hosts, YASIO_ARRAYSIZE(hosts));

    // for application protocol with length field, you just needs set this option.
    // it's similar to java netty length frame based decode.
    // such as when your protocol define as following
    //    packet.header: (header.len=12bytes)
    //            code:int16_t
    //            datalen:int32_t (not contains packet.header.len)
    //            timestamp:int32_t
    //            crc16:int16_t
    //    packet.data
    service->set_option(YOPT_C_LFBFD_PARAMS,
                        0,     // channelIndex, the channel index
                        65535, // maxFrameLength, max packet size
                        2,     // lenghtFieldOffset, the offset of length field
                        4,     // lengthFieldLength, the size of length field, can be 1,2,4
                        12,    // lengthAdjustment if the value of length feild ==
packet.header.len + packet.data.len, this parameter should be 0, otherwise should be
sizeof(packet.header)
    );

    // for application protocol without length field, just sets length field size to -1.
    // then io_service will dispatch any packet received from server immediately,
    // such as http request, this is default behavior of channel.
    service->set_option(YOPT_C_LFBFD_PARAMS, 1, 65535, -1, 0, 0);
    return 0;
}
```

See also

io_event Class

io_channel Class

Provides the functionality of establishing tcp/udp/kcp connections.

Syntax

```
namespace yasio { namespace inet { class io_channel; } }
```

Public Methods

Name	Description
io_channel::get_service	Gets belong service of channel.
io_channel::index	Gets index of channel at service.
io_channel::remote_port	Gets remote port of channel.

Remarks

Once io_service initialized, the max count of channel can't be changed.
Retrieves through `io_service::channel_at`.

io_channel::get_service

Gets owner service.

```
io_service& get_service()
```

io_channel::index

Gets channel index at service.

```
int index() const
```

io_channel::remote_port

Gets remote port.

```
u_short remote_port() const;
```

Return value

Return remote port of channel

- For client channel, it's port to connect.
- For server channel, it's port to listen.

See also

io_service Class

io_event Class

io_event Class

The event produced by io_service thread.

Syntax

```
namespace yasio { namespace inet { class io_event; } }
```

Public Methods

Name	Description
io_event::kind	Gets kind of event.
io_event::status	Gets status of event.
io_event::packet	Gets packet of event.
io_event::timestamp	Gets timestamp of event.
io_event::transport	Gets transport of event.
io_event::transport_id	Gets transport id of event.
io_event::transport_udata	Gets/Sets transport user data.

.. _kind:

io_event::kind

Gets kind of event.

```
int kind() const;
```

Return value

Return the kind value, can be follow values

- `YEK_PACKET` : Packet event
- `YEK_CONNECT_RESPONSE` : Connect response event
- `YEK_CONNECTION_LOST` : Connection lost event

io_event::status

Gets the status of event.

```
int status() const;
```

Return Value

- 0: No error
- NZ: error occured, user only needs print the error status code.

io_event::packet

Gets packet of event.

```
std::vector<char>& packet()
```

Return value

Return the mutable reference to packet of event, user can use std::move to move it.

io_event::timestamp

Get timestamp in microseconds of event.

```
highp_time_t timestamp() const;
```

Return value

Return the timestamp in macroseconds.

io_event::transport_id

Gets transport unique id.

```
unsigned int transport_id() const;
```

Return Value

Return a unique id range in 32 bit uint.

io_event::transport_udata

Sets or Gets transport userdata.

```
template<typename _Uty>
_Uty io_event::transport_udata();

template<typename _Uty>
void io_event::transport_udata(_Uty uservalue);
```

Remark

User should manage the gc of userdata, such as:

- Store userdata when receive connect success event.
- Cleanup the userdata when receive connection lost.

See also

io_service Class

io_channel Class

obstream Class

Provides the functionality of Binary Writer.

Syntax

```
namespace yasio {
using obstream = basic_obstream<endian::network_convert_tag>;

// The fast binary writer without byte order convertion.
using fast_obstream = basic_obstream<endian::host_convert_tag>;
}
```

Members

Public Constructors

Name	Description
obstream::obstream	Constructs a `obstream` object.

Public Methods

Name	Description
obstream::write	Function template, write number value.
obstream::write_ix	Function template, write **7bit Encoded Int/Int64**.
obstream::write_v	Write blob data with **7bit Encoded Int lenght field**.
obstream::write_byte	Write 1 byte.
obstream::write_bytes	Write blob data without length field.
obstream::empty	Check is stream empty.

Name	Description
obstream::data	Retrieves stream data pointer.
obstream::length	Retrieves size of stream.
obstream::buffer	Retrieves the buffer object of the stream.
obstream::save	Save the stream binary data to file.

Remarks

When write int16~int64 and float/double, will auto convert host byte order to network byte order.

Requirements

Header: obstream.hpp

obstream::obstream

Constructs a `obstream` object.

```
obstream(size_t capacity = 128);

obstream(const obstream& rhs);

obstream(obstream&& rhs);
```

Example

TODO:

obstream::write

Write number value to stream with byte order convertion.

```
template<typename _Nty>
void obstream::write(_Nty value);
```

Parameters

value
The value to be written.

Remarks

The type *_Nty* of value could be any (1~8bytes) integral or float types

Example

TODO:

obstream::write_ix

Write 7Bit Encoded Int compressed value.

```
template<typename _Intty>
void obstream::write_ix(_Intty value);
```

Parameters

value
The value to be written.

Remarks

The type *_Intty* of value must be one of follows

- int32_t
- int64_t

This function behavior is compatible with dotnet

- BinaryWriter.Write7BitEncodedInt
- BinaryWriter.Write7BitEncodedInt64

Example

TODO:

obstream::write_v

Write blob data with 7Bit Encoded Int length field.

```
void write_v(cxx17::string_view sv);
```

Parameters

sv
The string_view value to be written.

Remarks

This function will write length field with 7Bit Encoded first, then call write_bytes to write the value.

Example

TODO:

obstream::write_byte

Write 1 byte to stream.

```
void write_byte(uint8_t value);
```

Parameters

value
The value to be written.

Remarks

This function is identical to obstream::write

Example

TODO:

obstream::write_bytes

Write byte array to stream.

```
void write_bytes(cxx17::string_view sv);

void write_bytes(const void* data, int length);

void write_bytes(std::streamoff offset, const void* data, int length);
```

Parameters

sv
The string_view value to be written.

data
The data to be written.

length
The length data to be written.

offset
The offset of stream to be written.

Remarks

The value of `offset + length` must be less than `obstream::length`

Example

TODO:

obstream::empty

Tests whether the obstream is empty.

```
bool empty() const;
```

Return Value

`true` if the obstream empty; `false` if it has at least one byte.

Remarks

The member function is equivalent to length == 0.

Example

TODO:

obstream::data

Retrieves stream data pointer.

```
const char* data() const;

char* data();
```

Return Value

A pointer to the first byte in the stream.

Example

TODO:

obstream::length

Returns the number of bytes in the stream.

```
size_t length() const;
```

Return Value

The current length of the stream.

Example

TODO:

obstream::buffer

Retrieves internal buffer of stream.

```
const std::vector<char>& buffer() const;

std::vector<char>& buffer();
```

Return Value

The internal implementation buffer of the stream.

Example

```
// obstream_buffer.cpp
// compile with: /EHsc
#include "yasio/obstream.hpp"

int main( )
{
   using namespace yasio;
   using namespace cxx17;

   obstream obs;
   obs.write_v("hello world!");

   const auto& const_buffer = obs.buffer();

   // after this line, the obs will be empty
   auto move_buffer = std::move(obs.buffer());

   return 0;
}
```

obstream::save

Save the stream data to file.

```
void save(const char* filename) const;
```

Example

```
// obstream_save.cpp
// compile with: /EHsc
#include "yasio/obstream.hpp"
#include "yasio/ibstream.hpp"

int main( )
{
   using namespace yasio;
   using namespace cxx17;

   obstream obs;
   obs.write_v("hello world!");
   obs.save("obstream_save.bin");

   ibstream ibs;
   if(ibs.load("obstream_save.bin")) {
```

```
        // output should be: hello world!
        try {
            std::count << ibs.read_v() << "\n";
        }
        catch(const std::exception& ex) {
            std::count << "read_v fail: " <<
                << ex.message() << "\n";
        }
    }

    return 0;
}
```

See also

ibstream_view Class

ibstream Class

ibstream_view Class

Provides the functionality of Binary Reader.

Syntax

```
namespace yasio {
using ibstream_view = basic_ibstream_view<endian::network_convert_tag>;
using fast_ibstream_view = basic_ibstream_view<endian::host_convert_tag>;
}
```

Members

Public Constructors

Name	Description
ibstream_view::ibstream_view	Constructs a `ibstream_view` object.

Public Methods

Name	Description
ibstream_view::reset	Reset input data, weak reference.
ibstream_view::read	Function template, read number value.
ibstream_view:read_ix	Function template,read **7bit Encoded Int/Int64**.
ibstream_view:read_v	Read blob data with **7bit Encoded Int/Int64 lenght field**.
ibstream_view:read_byte	Read 1 byte.
ibstream_view:read_bytes	Read blob data without length field.
ibstream_view::empty	Check is stream empty.

Name	Description
ibstream_view::data	Retrieves stream data pointer.
ibstream_view::length	Retrieves size of stream.
ibstream_view::seek	Moves the read position in a stream.

Remarks

This class is inspired from C++17 std::string_view, it never copy any buffer during initialize and read.

Requirements

Header: ibstream.hpp

ibstream_view::ibstream_view

Constructs a `ibstream_view` object.

```
ibstream_view();

ibstream_view(const void* data, size_t size);

ibstream_view(const obstream* obs);
```

Parameters

data
The pointer to first byte of buffer.

size
The size of data.

obs
The obstream object.

Example

TODO:

ibstream_view::reset

Resets `ibstream_view` input buffer view.

```
void ibstream_view::reset(const void* data, size_t size);
```

Parameters

data
The pointer to first byte of buffer.

size
The size of data.

ibstream_view::read

Read number value from stream with byte order convertion.

```
template<typename _Nty>
_Nty ibstream_view::read();
```

Return Value

Returns the value to be read.

Remarks

The type *_Nty* of value could be any (1~8bytes) integral or float types.

Example

TODO:

ibstream_view::read_ix

Read 7Bit Encoded Int compressed value.

```
template<typename _Intty>
_Intty ibstream_view::read_ix();
```

Return Value

Returns the value to be read.

Remarks

The type *_Intty* of value must be one of follows

- int32_t
- int64_t

This function behavior is compatible with dotnet

- BinaryReader.Read7BitEncodedInt()
- BinaryReader.Read7BitEncodedInt64()

Example

TODO:

ibstream_view::read_v

Read blob data with 7Bit Encoded Int length field.

```
cxx17::string_view read_v();
```

Return Value

Returns the blob view to be read

Remarks

This function will read length field with 7Bit Encoded first, then call read_bytes to read the value.

Example

TODO:

ibstream_view::read_byte

Read 1 byte from stream.

```
uint8_t read_byte();
```

Return Value

Returns the value to be read.

Remarks

This function is identical to ibstream_view::read

Example

TODO:

ibstream_view::read_bytes

Read byte array from stream.

```
cxx17::string_view read_bytes();
```

Return Value

The blob view to be read.

Example

TODO:

ibstream_view::empty

Tests whether the ibstream_view is empty.

```
bool empty() const;
```

Return Value

`true` if the ibstream_view empty; `false` if it has at least one byte.

Remarks

The member function is equivalent to length == 0.

Example

TODO:

ibstream_view::data

Retrieves stream data pointer.

```
const char* data() const;
```

Return Value

A pointer to the first byte in the stream.

Example

TODO:

ibstream_view::length

Returns the number of bytes in the stream.

```
size_t length() const;
```

Return Value

The current length of the stream.

Example

TODO:

ibstream_view::seek

Moves the read position in a stream.

```
ptrdiff_t seek(ptrdiff_t offset, int whence);
```

Parameters

offset\ An offset to move the read pointer relative to *whence*.

whence\ One of the `SEEK_SET`, `SEEK_CUR`, `SEEK_END` enumerations.

Return Value

The current read poistion of the stream after seek.

Example

TODO:

ibstream Class

Provides the functionality of Binary Reader with buffer storage.

Syntax

```
namespace yasio {
using ibstream = basic_ibstream<endian::network_convert_tag>;
using fast_ibstream = basic_ibstream<endian::host_convert_tag>;
}
```

Members

Public Constructors

Name	Description
ibstream::ibstream	Constructs a `ibstream` object.

Public Methods

Name	Description
ibstream::load	Load stream from file.

Inheritance Hierarchy

ibstream_view

`ibstream`

ibstream::ibstream

Constructs a `ibstream` object.

```
ibstream(std::vector<char> blob);

ibstream(const obstream* obs);
```

Parameters

blob
The input binary buffer.

obs
The obstream object.

Example

TODO:

ibstream::load

Load the stream data from file.

```
bool load(const char* filename) const;
```

Return Value

`true` succed, `false` fail.

Example

See: obstream::save

See also

obstream Class

io_service Class

xxsocket Class

Provides the functionality of low-level socket based on POSIX socket APIs, support std::move

Syntax

```
namespace yasio { namespace inet { class xxsocket; } }
```

Members

Name	Description
xxsocket::xxsocket	Constructs a `xxsocket` object.

Public Methods

Name	Description
xxsocket::xpconnect	Cnnect remote via tcp.
xxsocket::xpconnect_n	Connect remote via tcp non-blocking.
xxsocket::pconnect	Connect remote via tcp.
xxsocket::pconnect_n	Connect remote via tcp non-blocking.
xxsocket::pserve	Create socket as tcp server.
xxsocket::swap	Swap socket handle.
xxsocket::open	Open a socket.
xxsocket::reopen	Reopen a socket.
xxsocket::is_open	Check whether socket opened.

Name	Description
xxsocket::native_handle	Gets socket handle.
xxsocket::release_handle	Release ownership of socket handle.
xxsocket::set_nonblocking	Sets socket non-blocking mode.
xxsocket::test_nonblocking	Test whether socket is non-blocking mode.
xxsocket::bind	Bind socket with specific address.
xxsocket::bind_any	Bind socket with any address.
xxsocket::listen	Listen a tcp socket.
xxsocket::accept	Accept a tcp socket.
xxsocket::accept_n	Accept a tcp socket non-blocking.
xxsocket::connect	Connect a socket.
xxsocket::connect_n	Connect a socket non-blocking.
xxsocket::send	Send data on the socket.
xxsocket::send_n	Send data on the socket non-blocking.
xxsocket::recv	Receive data from the socket.
xxsocket::recv_n	Receive data from the socket non-blocking.
xxsocket::sendto	Send data to a DGRAM socket.
xxsocket::recvfrom	Send data to a DGRAM socket non-blocking.
xxsocket::handle_write_ready	Wait socket ready to write.
xxsocket::handle_read_ready	Wait socket ready to read.

Name	Description
xxsocket::local_endpoint	Gets local endpoint of socket.
xxsocket::peer_endpoint	Gets peer endpoint of socket.
xxsocket::set_keepalive	Sets tcp socket keepalive.
xxsocket::reuse_address	Sets socket reuse address.
xxsocket::exclusive_address	Sets socket exclusive address.
xxsocket::select	Select event ready for socket.
xxsocket::shutdown	Shutdown socket.
xxsocket::close	Close socket.
xxsocket::tcp_rtt	Gets tcp socket rtt.
xxsocket::get_last_errno	Gets last socket error.
xxsocket::set_last_errno	Sets last socket error.
xxsocket::strerror	Translate socket error code to string.
xxsocket::gai_strerror	Translate getaddrinfo error code to string.
xxsocket::resolve	Resolve domain.
xxsocket::resolve_v4	Resolve domain ipv4 address.
xxsocket::resolve_v6	Resolve domain ipv6 address.
xxsocket::resolve_v4to6	Resolve ipv4 address and convert to ipv6 V4MAPPED format.
xxsocket::resolve_tov6	Resolve all address, convert ipv4 address to ipv6 V4MAPPED format.
xxsocket::getipsv	Get local supported ip stack flags.

Name	Description
xxsocket::traverse_local_address	Traverse local address.

See also

io_service Class

io_service options

The following are the io_service options.

Name	Description
YOPT_S_DEFER_EVENT_CB	Set defer event callback params: callback:defer_event_cb_t remarks: a. User can do custom packet resolve at network thread, such as decompress and crc check. b. Return true, io_service will continue enque to event queue. c. Return false, io_service will drop the event.
YOPT_S_DEFERRED_EVENT	Set whether deferred dispatch event, default is: 1 params: deferred_event:int(1)
YOPT_S_RESOLV_FN	Set custom resolve function, native C++ ONLY params: func:resolv_fn_t*
YOPT_S_PRINT_FN	Set custom print function native C++ ONLY parmas: func:print_fn_t remarks: you must ensure thread safe of it
YOPT_S_PRINT_FN2	Set custom print function with log level parmas: func:print_fn2_t you must ensure thread safe of it
YOPT_S_EVENT_CB	Set event callback params: func:event_cb_t*
YOPT_S_TCP_KEEPALIVE	Set tcp keepalive in seconds, probes is tries. params: idle:int(7200), interal:int(75), probes:int(10)
YOPT_S_NO_NEW_THREAD	Don't start a new thread to run event loop. params: value:int(0)
YOPT_S_SSL_CACERT	Sets ssl verification cert, if empty, don't verify. params: path:const char*

Name	Description
YOPT_S_CONNECT_TIMEOUT	Set connect timeout in seconds. params: connect_timeout:int(10)
YOPT_S_DNS_CACHE_TIMEOUT	Set dns cache timeout in seconds. params: dns_cache_timeout : int(600),
YOPT_S_DNS_QUERIES_TIMEOUT	Set dns queries timeout in seconds, default is: 5000. params: dns_queries_timeout : int(5000) remark: a. this option must be set before 'io_service::start' b. only works when have c-ares c. since v3.33.0 it's milliseconds, previous is seconds. d. the timeout algorithm of c-ares is complicated, usually, by default, dns queries will failed with timeout after more than 75 seconds. e. for more detail, please see: https://c-ares.haxx.se/ares_init_options.html
YOPT_S_DNS_QUERIES_TRIES	Set dns queries tries when timeout reached, default is: 5. params: dns_queries_tries : int(5) remarks: a. this option must be set before 'io_service::start' b. relative option: *YOPT_S_DNS_QUERIES_TIMEOUT*
YOPT_S_DNS_DIRTY	Set dns server dirty. params: reserved : int(1) remarks: a. this option only works with c-ares enabled b. you should set this option after your mobile network changed
YOPT_C_LFBFD_FN	Sets channel length field based frame decode function. params: index:int, func:decode_len_fn_t* remark: native C++ ONLY
YOPT_C_LFBFD_PARAMS	Sets channel length field based frame decode params. params: index:int, max_frame_length:int(10MBytes), length_field_offset:int(-1), length_field_length:int(4), length_adjustment:int(0),

Name	Description
YOPT_C_LFBFD_IBTS	Sets channel length field based frame decode initial bytes to strip. params:index:int,initial_bytes_to_strip:int(0)
YOPT_C_REMOTE_HOST	Sets channel remote host. params: index:int, ip:const char*
YOPT_C_REMOTE_PORT	Sets channel remote port. params: index:int, port:int
YOPT_C_REMOTE_ENDPOINT	Sets channel remote endpoint. params: index:int, ip:const char*, port:int
YOPT_C_LOCAL_HOST	Sets local host for client channel only. params: index:int, ip:const char*
YOPT_C_LOCAL_PORT	Sets local port for client channel only. params: index:int, port:int
YOPT_C_LOCAL_ENDPOINT	Sets local endpoint for client channel only. params: index:int, ip:const char*, port:int
YOPT_C_MOD_FLAGS	Mods channl flags. params: index:int, flagsToAdd:int, flagsToRemove:int
YOPT_C_ENABLE_MCAST	Enable channel multicast mode. params: index:int, multi_addr:const char*, loopback:int
YOPT_C_DISABLE_MCAST	Disable channel multicast mode. params: index:int
YOPT_C_KCP_CONV	The kcp conv id, must equal in two endpoint from the same connection. params: index:int, conv:int
YOPT_T_CONNECT	Change 4-tuple association for io_transport_udp. params: transport:transport_handle_t remark: only works for udp client transport
YOPT_T_DISCONNECT	Dissolve 4-tuple association for io_transport_udp. params: transport:transport_handle_t remark: only works for udp client transport

Name	Description
YOPT_B_SOCKOPT	Sets io_base sockopt. params: io_base*,level:int,optname:int,optval:int,optlen:int

See also

io_service Class

yasio Macros

The macros listed in the table below may be used to control the interface, functionality, and behaviour of `yasio`. You can define them at yasio/detail/config.hpp or compiler preprocessors.

Name	Description
YASIO_HAVE_KCP	Whether enable kcp, default: `off`
YASIO_HEADER_ONLY	Whether enable header only, default: `off`
YASIO_SSL_BACKEND	Choose ssl backend, since 3.36.0 `1`. Use OpenSSL `2`. Use mbedtls
YASIO_ENABLE_UDS	Whether enable unix domain socket support, current only unix-like system and win10 RS5 support this feature, default: `off`
YASIO_HAVE_CARES	Whether use c-ares to resolve domain name, default: `off`
YASIO_VERBOSE_LOG	Whether enable verbose log, default: `off`
YASIO_NT_COMPAT_GAI	Whether enable windows xp `getaddrinfo` API compatible, default: `off`
YASIO_USE_SPSC_QUEUE	Whether use SPSC queue, default: `off`
YASIO_HAVE_HALF_FLOAT	Whether enable half float, depends on half.hpp
YASIO_DISABLE_OBJECT_POOL	Whether disable object pool
YASIO_DISABLE_CONCURRENT_SINGLETON	Whether disable concurrent singleton

FAQ

FAQ

Can't load xlua bundle on macOS?

The file `xlua.bundle` needs change attr by command `sudo xattr -r -d com.apple.quarantine xlua.bundle`