© 2021 yasio.org

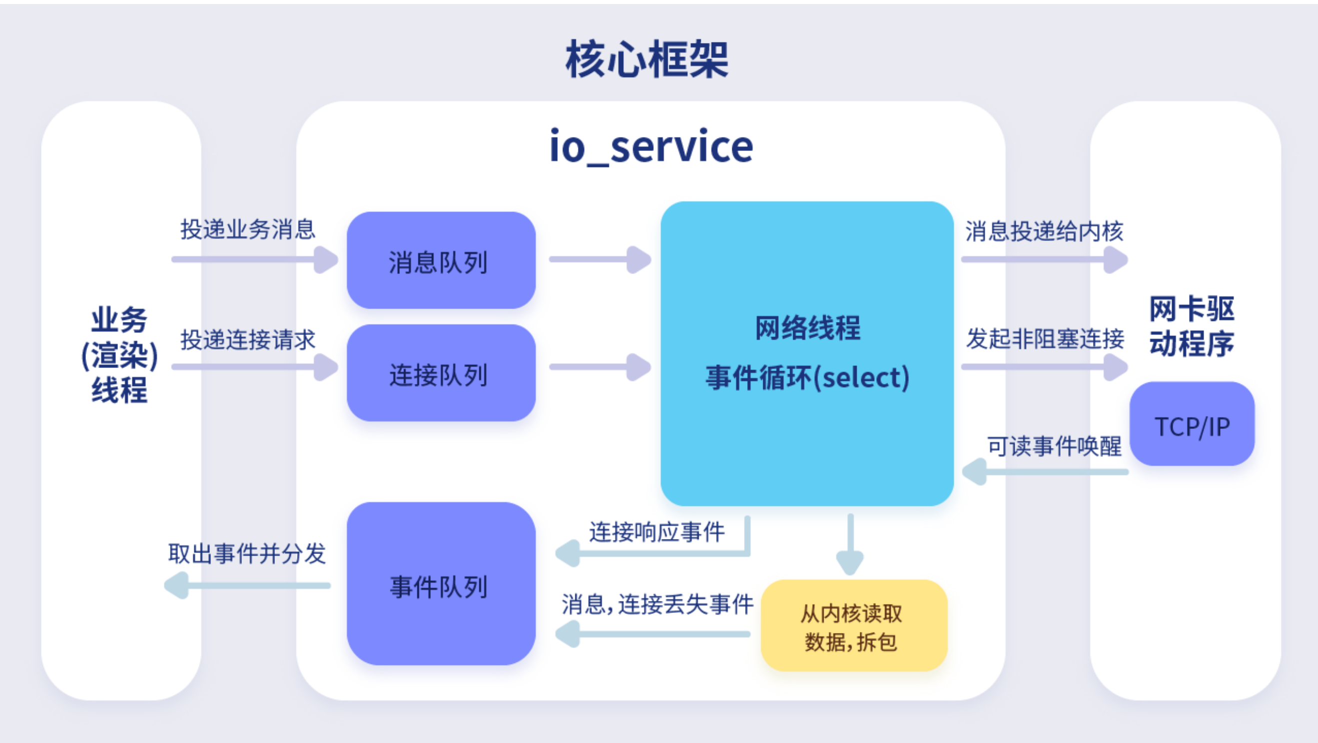
© 2021 yasio.org

yasio 中文文档

docs passing

yasio 是一个轻量级跨平台的异步socket库，专注于客户端和基于各种游戏引擎的游戏客户端网络服务。

- 跨平台性:
    - 编译器:
        - Visual Studio 2013+
        - GCC4.7+
        - xcode9+
        - 全面支持 C++11,14,17 的编译器
    - 架构: x86, x64, ARM等。
    - 操作系统: Windows, macOS, Linux, FreeBSD, iOS, Android等。
- 术语:
    - 网络服务: `io_service`
    - 信道: `io_channel`
    - 传输会话: `io_transport`
- 框架图:



© 2021 yasio.org

5F905F59
EB1F00CB

6B5B4F7A5E7B5354 `tool.chinaz.com` 5301http8BF76C5E625354CD5E65637002

© 2021 yasio.org

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_PACKET: {
            auto packet = std::move(ev->packet());
            fwrite(packet.data(), packet.size(), 1, stdout);
            fflush(stdout);
            break;
        }
        case YEK_CONNECT_RESPONSE:
            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_CONNECTION_LOST:
            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 = ""
```

© 2021 yasio.org

```lua
service:start(function(ev)
        local k = ev.kind()
        if (k == yasio.YEK_PACKET) then
            respdata = respdata .. ev:packet():to_string()
        elseif k == yasio.YEK_CONNECT_RESPONSE 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_CONNECTION_LOST 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
```

测试 & 示例

> **注意**
>
> 运行Lua示例程序，会打印异常信息 `yasio - ibstream_view::consume out of range!` ，这是示例程序里故意写的，请不必在意。

- 测试:
  - echo_server: TCP/UDP/KCP 回声服务器
  - echo_client: TCP/UDP/KCP 回声客户端
  - ssltest: SSL测试客户端, 请求github.com主页并打印响应数据
  - tcptest: TCP测试程序

© 2021 yasio.org

    - speedtest: TCP,UDP,KCP
    - mcast:
- :
    - ftp_server: yasioftp
    - lua: Luahttp TCP
    - xlua: xlua
    - DemoUE4: UE4

&

- C++11 `msvc`, `gcc`, `clang`
- `git`, `cmake` installed
- :

```
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
```

© 2021 yasio.org

© 2021 yasio.org

io_service Class

socket.select tcp, udp, kcp, ssl-client


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


Name	Description
io_service::io_service	io_service

Name	Description
io_service::start	
io_service::stop	
io_service::open	
io_service::close	
io_service::is_open	
io_service::dispatch	
io_service::write	
io_service::write_to	DGRAM

© 2021 yasio.org

Name	Description
io_service::schedule	6C 51 58 65 50 E8 8C 9A F6 68
io_service::init_globals	66 79 52 59 53 51 5C 65 63 3E 3A 1D CB 16 68 40 70 6E
io_service::cleanup_globals	6E 74 51 5C 65 63 05 68 68 40 70 6E
io_service::channel_at	83 53 4F 90 53 67 B7 D6 E1 53 E5 C4
io_service::set_option	8B 7E 90 98 BE 6E 69 79

6C 61
E8 0F

9E 8B 4F 8F 4F 8B 76 52 5E 4F 4F 75 58 8C 6C D8 A4 20 93 1A DD 84 1B FA 1A 7F 28 F9 61 60 `object_pool` 30 02

89 6C
81 42

59 65 4E 34 87 F6: yasio.hpp

io_service::io_service

67 90 84 28 `io_service` 5B 8C 39 F9 61 02

```
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);
```

53 65
C2 70

channel_count
4F 99 65 91 39 E1 53 78 CF 02

channel_ep
4F 99 8F 7A 57 57 39 E1 53 DC EF 30 40 02

© 2021 yasio.org

channel_eps

[illegible]

[illegible]

```
#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

[illegible]

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

[illegible]

cb

[illegible] io_service::dispatch [illegible]

[illegible]

```
#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());
    }
    });
```

© 2021 yasio.org

```
    return 0;
}
```

io_service::stop

```
void stop()
```

io_service::open

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

cindex

kind

TCP, io_service:

kind :

- `YCK_TCP_CLIENT`
- `YCK_TCP_SERVER`
- `YCK_UDP_CLIENT`
- `YCK_UDP_SERVER`
- `YCK_KCP_CLIENT`
- `YCK_KCP_SERVER`

© 2021 yasio.org

- `YCK_SSL_CLIENT`

io_service::close

[illegible]

```
void close(transport_handle_t transport);

void close(int cindex);
```

[illegible]

transport

[illegible]

cindex

[illegible]

[illegible]

[illegible] `TCP` [illegible]4[illegible]

io_service::is_open

[illegible]

```
bool is_open(transport_handle_t transport) const;

bool is_open(int cindex) const;
```

[illegible]

transport

[illegible]

cindex

[illegible]

[illegible]

`true`: [illegible] `false`: [illegible]

© 2021 yasio.org

io_service::dispatch

[illegible]

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

[illegible]

max_count

[illegible]

[illegible]

[illegible], [illegible]Cocos2d-x[illegible](Unity,UE4)[illegible]

[illegible]

[illegible]

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

io_service::write

[illegible]

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

[illegible]

thandle

[illegible]

buffer

[illegible]

completion_handler

[illegible]

© 2021 yasio.org

返回值

返回发送数据字节数, `< 0`: 说明发生错误。

注意

completion_handler 不支持 `KCP` 。

空buffer会直接被忽略，也不会触发 *completion_handler* 。

io_service::write_to

向UDP传输会话发送数据。

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

参数

thandle
传输会话句柄。

buffer
要发送的buffer。

to
要发送的远端地址。

completion_handler
发送完成回调。

返回值

成功发送的字节数, 当 `< 0` 时说明发生错误，通常是传输会话已关闭(TCP连接断开等)。

注意

此函数仅可用于 *DGRAM* 传输会话，即 `UDP,KCP` 。

发送完成回调 *completion_handler* 不支持 `KCP` 。

© 2021 yasio.org

buffer *completion_handler*

io_service::schedule

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

duration

cb

`std::shared_ptr`

```
// 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

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

© 2021 yasio.org

print_fn

```
// 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

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

© 2021 yasio.org

注意

当用户需要卸载包含自定义日志打印回调的动态库(.dll,.so)前必须调用此函数，谨防应用程序闪退。

io_service::channel_at

通过信道索引获取信道句柄。

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

参数

cindex
信道索引

返回值

信道句柄指针, 当索引超出范围时，返回 `nullptr` 。

io_service::set_option

设置选项。

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

参数

opt
选项枚举, 请查看 YOPT_X_XXX.

示例

```
#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));
```

© 2021 yasio.org

```
    // 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;
}
```

8B 53 96
F7 C2 05

© 2021 yasio.org

© 2021 yasio.org

io_channel Class

TCP/SSL/UDP/KCP


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


Name	Description
io_channel::get_service	io_service
io_channel::index	
io_channel::remote_port	
io_channel::bytes_transferred	
io_channel::connect_id	ID

io_service
`io_service::channel_at`

io_channel::get_service

io_service

```
io_service& get_service()
```

© 2021 yasio.org

io_channel::index

[illegible]

```
int index() const
```

io_channel::remote_port

[illegible].

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

[illegible]

[illegible]

- [illegible]
- [illegible]

io_channel::bytes_transferred

[illegible]

```
long long bytes_transferred() const;
```

[illegible]

[illegible]

io_channel::connect_id

[illegible]id

```
unsigned int connect_id() const;
```

[illegible]

[illegible]id

© 2021 yasio.org

8B 53 96
F7 C2 05

io_service Class

io_event Class

© 2021 yasio.org

© 2021 yasio.org

io_event Class

io_service

8B6C
ED D5

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

62 54
10 58

51 51 65 6C
6C 71 B9 D5

Name	Description
io_event::kind	
io_event::status	
io_event::packet	
io_event::timestamp	
io_event::transport	
io_event::transport_id	ID
io_event::transport_udata	

io_event::kind

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

© 2021 yasio.org

8F 56 50
D4 DE 3C

4E 4E 7C 57 FF 53 4E 66 4E 4E 50:

- `YEK_PACKET` : 60 69 53 4E 4E
- `YEK_CONNECT_RESPONSE` : 8E 63 54 5E 4E 4E
- `YEK_CONNECTION_LOST` : 8E 63 4E 59 4E 4E

io_event::status

83 53 4E 4E 72 60 30
B7 D6 8B F6 66 01 02

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

8F 56 50
D4 DE 3C

- 0: 6B 5E
- 97 0: 51 95, 75 62 53 97 89 7B 53 62 53 53 30
  5E FA 19 28 37 EA 00 81 80 55 53 70 73 EF 02

io_event::packet

83 53 4E 4E 64 5E 76 6D 60 53
B7 D6 8B F6 3A 26 84 88 6F 05

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

8F 56 50
D4 DE 3C

6D 60 53 76 5E 75, 75 62 53 4E 4F 75 `std::move` 65GC
88 6F 05 84 15 28, 28 37 EF E5 7F 28 E0 B9 0F CE 8B F6 D6 70 88 6F 05 02

io_event::timestamp

83 53 4E 4E 75 76 5F 79 7E 65 95 62 30
B7 D6 8B F6 A7 1F 84 AE D2 A7 F6 F4 33 02

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

8F 56 50
D4 DE 3C

54 7C 7E 65 95 65 51 76 5E 79 7E 65 95 62 30
8C FB 0F F6 F4 E0 73 84 AE D2 A7 F6 F4 33 02

© 2021 yasio.org

io_event::transport

[illegible]

```
transport_handle_t transport() const;
```

[illegible]

[illegible]

io_event::transport_id

[illegible]ID[illegible]

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

[illegible]

32[illegible]ID[illegible]

io_event::transport_udata

[illegible]

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

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

[illegible]

[illegible] userdata [illegible], [illegible]:

- [illegible]userdata
- [illegible]userdata

[illegible]

io_service Class

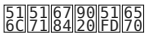
© 2021 yasio.org

© 2021 yasio.org

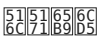
© 2021 yasio.org

obstream Class

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



Name	Description
obstream::obstream	1 `obstream`



Name	Description
obstream::write	
obstream::write_ix	(**7bit Encoded Int/Int64**)
obstream::write_v	(**7bit Encoded Int**)
obstream::write_byte	1

© 2021 yasio.org

Name	Description
obstream::write_bytes	写入字节集
obstream::empty	是否为空
obstream::data	获取数据
obstream::length	获取长度
obstream::buffer	获取缓冲区
obstream::clear	清空
obstream::shrink_to_fit	收缩到合适
obstream::save	保存到文件

成员

头文件: obstream.hpp

obstream::obstream

构造 obstream 对象

```
obstream(size_t capacity = 128);

obstream(const obstream& rhs);

obstream(obstream&& rhs);
```

obstream::write

写入数据

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

© 2021 yasio.org

value

_Nty 1~8

obstream::write_ix

32/64 7Bit Encoded Int

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

value

_Intty

- int32_t
- int64_t

dotnet

- BinaryWriter.Write7BitEncodedInt
- BinaryWriter.Write7BitEncodedInt64

obstream::write_v

(7Bit Encoded Int).

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

© 2021 yasio.org

sv

data

7Bit Encoded, write_bytes .

obstream::write_byte

1

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

value

obstream::write

obstream::write_bytes

```
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);
```

sv

string_view.

data

.

© 2021 yasio.org

length

[illegible].

offset

[illegible].

[illegible]

`offset + length` [illegible] `obstream::length`

obstream::empty

[illegible]

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

[illegible]

`true` [illegible]; `false` [illegible]

[illegible]

[illegible] length == 0[illegible]

obstream::data

[illegible]

```
const char* data() const;

char* data();
```

[illegible]

[illegible]

obstream::length

[illegible]

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

© 2021 yasio.org

8F5650
D4DE3C

8F56604F5354766058825530
D4DE412D05288438573270 02

obstream::buffer

83536051900F5153 30
B7D64185E8132A 02

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

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

8F5650
D4DE3C

60519 0F5153 5E75FF534F75
4185E8132A155 28CEF5E7F28 `std::move` 53800
D870 02

794F
3A8B

```
// 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::clear

6E746DFF4E4F5975 30
65064160CE5BF6028 02

```
void clear();
```

© 2021 yasio.org

注意

此函数不会释放buffer内存，对于高效地复用序列化器非常有用。

obstream::shrink_to_fit

释放流内部缓冲区多余内存。

```
void shrink_to_fit();
```

obstream::save

将流中的二进制字节数据保存到文件。

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

示例

```
// 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;
}
```

© 2021 yasio.org

8B 53 96
F7 C2 05

ibstream_view Class

ibstream Class

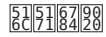
© 2021 yasio.org

© 2021 yasio.org

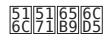
ibstream_view Class

```
namespace yasio {
// 
using ibstream_view = basic_ibstream_view<endian::network_convert_tag>;

// 
using fast_ibstream_view = basic_ibstream_view<endian::host_convert_tag>;
}
```



Name	Description
ibstream_view::ibstream_view	1 `ibstream_view`



Name	Description
ibstream_view::reset	
ibstream_view::read	
ibstream_view:read_ix	**(7bit Encoded Int/Int64)**
ibstream_view:read_v	**(7bit Encoded Int/Int64)**

© 2021 yasio.org

Name	Description
ibstream_view:read_byte	读取1个字节
ibstream_view:read_bytes	读取指定长度二进制数据
ibstream_view::empty	检查流是否为空
ibstream_view::data	获取流数据指针
ibstream_view::length	获取流大小
ibstream_view::advance	向前移动流的读取游标
ibstream_view::seek	移动流的读取游标

注意

`ibstream_view` 借鉴C++17标准的 `std::string_view`, 意味着在初始化和反序列化过程中不会产生任何GC。

要求

头文件: ibstream.hpp

ibstream_view::ibstream_view

构造一个 `ibstream_view` 对象。

```
ibstream_view();

ibstream_view(const void* data, size_t size);

ibstream_view(const obstream* obs);
```

参数

data
待反序列化二进制数据首地址。

© 2021 yasio.org

size

5F 53 5E 52 53 4E 8F 52 65 63 59 5C 30 85 CD 8F 17 16 8C DB 36 70 6E 27 0F 02

obs

50 5E 52 53 76 60 39 F2 8F 17 16 84 41 02

ibstream_view::reset

91 7E `ibstream_view` 7F 51 89 56 30 CD 6E 13 B2 C6 FE 02

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

53 65 C2 70

data

5F 53 5E 52 53 4E 8F 52 65 63 99 57 57 30 85 CD 8F 17 16 8C DB 36 70 6E 96 30 40 02

size

5F 53 5E 52 53 4E 8F 52 65 63 59 5C 30 85 CD 8F 17 16 8C DB 36 70 6E 27 0F 02

ibstream_view::read

4E 60 4E 8B 53 65 50 30 CE 41 2D FB D6 70 3C 02

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

8F 56 50 D4 DE 3C

8F 56 8B 52 76 50 D4 DE FB 30 84 3C

6C 61 E8 0F

_Nty 5B 96 7C 57 53 4E 66 4E 61 9E 45 7B 8B EF E5 2F FB 0F 1~8 5B 82 65 65 7C 57 57 82 74 70 78 8B 62 6D 70 7C 57 16 6E B9 7B 8B 02

ibstream_view::read_ix

8B 53 FB D6 7Bit Encoded Int 53 7F 7F 78 76 65 50 BB 29 16 61 84 74 70 3C 02

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

© 2021 yasio.org

8F5650
D4DE3C

32/64

6C61
E80F

_Intty

- int32_t
- int64_t

Microsoft dotnet

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

ibstream_view::read_v

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

8F5650
D4DE3C

GC

6C61
E80F

7bit Encoded Int read_bytes

ibstream_view::read_byte

1

```
uint8_t read_byte();
```

8F5650
D4DE3C

uint8_t

© 2021 yasio.org

注意

本函数等价于 ibstream_view::read

ibstream_view::read_bytes

读取指定长度字节数据，无GC。

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

返回值

二进制数据的 `cxx17::string_view` 类型视图。

ibstream_view::empty

判断流是否为空。

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

返回值

`true` 空; `false` 流中至少包含1个字节。

注意

此方法等价于 length == 0。

ibstream_view::data

返回流数据指针。

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

返回值

返回指向流中第一个字节的指针。

© 2021 yasio.org

ibstream_view::length

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

ibstream_view::advance

```
void advance(ptrdiff_t offset);
```

offset

ibstream_view::seek

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

offset

whence

SEEK_SET , SEEK_CUR , SEEK_END

© 2021 yasio.org

ibstream Class

8B6CED D5

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

62541058

51516790516571847028F070

Name	Description
ibstream::ibstream	67901C4E2A ibstream 5B8CF961

51516C71B9D5

Name	Description
ibstream::load	4E654E528F6DCE87FE8A07D41

7E625C6B7E67E77F4221D384

ibstream_view

ibstream

ibstream::ibstream

67902C4E4E2A ibstream 5B8C30F96162

© 2021 yasio.org

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

ibstream(const obstream* obs);
```

[illegible]

blob

[illegible]

obs

[illegible] `obstream` [illegible]

ibstream::load

[illegible].

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

[illegible]

`true` [illegible] `false` [illegible]

[illegible]

[illegible] obstream::save

[illegible]

obstream Class

io_service Class

© 2021 yasio.org

© 2021 yasio.org

xxsocket Class

bsd socket API yasio

xxsocket `accept_n` `xxx_n` socket

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

Name	Description
xxsocket::xxsocket	`xxsocket`

Name	Description
xxsocket::xpconnect	TCP
xxsocket::xpconnect_n	TCP
xxsocket::pconnect	TCP
xxsocket::pconnect_n	TCP
xxsocket::pserve	tcp
xxsocket::swap	socket
xxsocket::open	socket

© 2021 yasio.org

Name	Description
xxsocket::reopen	重新打开socket
xxsocket::is_open	检查socket是否已经打开
xxsocket::native_handle	获取socket句柄
xxsocket::release_handle	释放socket句柄所有权
xxsocket::set_nonblocking	设置socket非阻塞模式
xxsocket::test_nonblocking	检测socket非阻塞模式
xxsocket::bind	绑定本地地址和端口
xxsocket::bind_any	绑定本地任意地址和端口
xxsocket::listen	开始监听TCP端口
xxsocket::accept	接受一个TCP连接
xxsocket::accept_n	非阻塞接受一个TCP连接
xxsocket::connect	连接
xxsocket::connect_n	非阻塞连接
xxsocket::send	发送
xxsocket::send_n	非阻塞发送
xxsocket::recv	接收
xxsocket::recv_n	非阻塞接收
xxsocket::sendto	发送DGRAM
xxsocket::recvfrom	接收DGRAM

© 2021 yasio.org

Name	Description
xxsocket::handle_write_ready	等待socket可写
xxsocket::handle_read_ready	等待socket可读
xxsocket::local_endpoint	获取socket本地地址
xxsocket::peer_endpoint	获取socket远端地址
xxsocket::set_keepalive	设置tcp keepalive
xxsocket::reuse_address	设置socket是否重用地址
xxsocket::exclusive_address	设置socket是否阻止地址重用
xxsocket::select	监听socket事件
xxsocket::shutdown	停止socket收发
xxsocket::close	关闭socket
xxsocket::tcp_rtt	获取tcp rtt.
xxsocket::get_last_errno	获取最近socket错误码
xxsocket::set_last_errno	设置最近socket错误码
xxsocket::strerror	将socket错误码转换为字符串
xxsocket::gai_strerror	将getaddrinfo返回值转换为字符串
xxsocket::resolve	解析域名
xxsocket::resolve_v4	解析域名包含的ipv4地址
xxsocket::resolve_v6	解析域名包含的ipv6地址
xxsocket::resolve_v4to6	解析域名包含的ipv4地址并转换为ipv6的V4MAPPED格式

© 2021 yasio.org

Name	Description
xxsocket::resolve_tov6	ipv4ipv6V4MAPPED
xxsocket::getipsv	ip
xxsocket::traverse_local_address	

xxsocket::xxsocket

 `xxsocket`

```
xxsocket::xxsocket();
xxsocket::xxsocket(socket_native_type handle);
xxsocket::xxsocket(xxsocket&& right);
xxsocket::xxsocket(int af, int type, int protocol);
```

handle
socket `xxsocket`

right
move

af
ip

protocol
TCP/UDP `0`

xxsocket::xpconnect

TCP

```
int xxsocket::xpconnect(const char* hostname, u_short port, u_short local_port = 0);
```

hostname
 IP

© 2021 yasio.org

port

local_port

`0`

hostname

IP

port

local_port

`0`

wtimeout

`0`

© 2021 yasio.org

xxsocket::pconnect

[illegible]TCP[illegible]

```
int xxsocket::pconnect(const char* hostname, u_short port, u_short local_port = 0);
int xxsocket::pconnect(const endpoint& ep, u_short local_port = 0);
```

[illegible]

hostname

[illegible] IP[illegible]

ep

[illegible]

port

[illegible]

local_port

[illegible] `0` [illegible]

[illegible]

`0`:[illegible] `< 0` [illegible] `xxsocket::get_last_errno` [illegible]

[illegible]

[illegible]ip[illegible]

xxsocket::pconnect_n

[illegible]TCP[illegible]

```
int pconnect_n(const char* hostname, u_short port, const std::chrono::microseconds& wtimeout,
u_short local_port = 0);
int pconnect_n(const char* hostname, u_short port, u_short local_port = 0);
int pconnect_n(const endpoint& ep, const std::chrono::microseconds& wtimeout, u_short
local_port = 0);
int pconnect_n(const endpoint& ep, u_short local_port = 0);
```

[illegible]

hostname

[illegible] IP[illegible]

© 2021 yasio.org

ep

port

local_port

`0` `xxsocket::get_last_errno`

wtimeout

xxsocket::pserve

TCP

```
int pserve(const char* addr, u_short port);
int pserve(const endpoint& ep);
```

addr

IP

ep

port

TCP

`0` `xxsocket::get_last_errno`

```
// xxsocket-serve.cpp
#include <signal.h>
#include <vector>
#include "yasio/xxsocket.hpp"
```

© 2021 yasio.org

```
using namespace yasio;
using namespace yasio::inet;

xxsocket g_server;
static bool g_stopped = false;
void process_exit(int sig)
{
  if (sig == SIGINT)
  {
    g_stopped = true;
    g_server.close();
  }
  printf("exit");
}
int main()
{
  signal(SIGINT, process_exit);

  if (g_server.pserve("0.0.0.0", 1219) != 0)
    return -1;
  const char reply_msg[] = "hi, I'm server\n";
  do
  {
    xxsocket cs = g_server.accept();
    if (cs.is_open())
    {
      cs.send(reply_msg, sizeof(reply_msg) - 1);
      std::this_thread::sleep_for(std::chrono::milliseconds(100));
    }
  } while (!g_stopped);

  return 0;
}
```

xxsocket::swap

交换两个socket对象

```
xxsocket& swap(xxsocket& who);
```

参数

who

要交换的对象

返回值

当前对象

`xxsocket` 对象

© 2021 yasio.org

xxsocket::open

socket

```
bool open(int af = AF_INET, int type = SOCK_STREAM, int protocol = 0);
```

af
AF_INET (ipv4) AF_INET6 (ipv6)

type
socket SOCK_STREAM (TCP), SOCK_DGRAM (UDP)

protocol
TCP/UDP 0

true : false xxsocket::get_last_errno

xxsocket::reopen

socket

```
bool reopen(int af = AF_INET, int type = SOCK_STREAM, int protocol = 0);
```

af
AF_INET (ipv4) AF_INET6 (ipv6)

type
socket SOCK_STREAM (TCP), SOCK_DGRAM (UDP)

protocol
TCP/UDP 0

true : false xxsocket::get_last_errno

© 2021 yasio.org

xxsocket::is_open

socket

```
bool is_open() const;
```

true : false :

xxsocket::native_handle

socket

```
socket_native_type native_handle() const;
```

socket yasio::inet::invalid_socket socket

xxsocket::release_handle

socket

```
socket_native_type release_handle() const;
```

socket

xxsocket::set_nonblocking

socket

```
int set_nonblocking(bool nonblocking) const;
```

© 2021 yasio.org

参数

nonblocking
`true`：非阻塞模式， `false`：阻塞模式。

返回值

`0`：成功， `< 0` 失败，通过 `xxsocket::get_last_errno` 获取错误码。

xxsocket::test_nonblocking

检测socket是否为非阻塞模式。

```
int test_nonblocking() const;
```

返回值

`1`：非阻塞模式， `0`：阻塞模式。

注意

对于winsock2，未连接的 `SOCK_STREAM` 类型socket会返回 `-1` 。

xxsocket::bind

绑定socket本机地址。

```
int bind(const char* addr, unsigned short port) const;
int bind(const endpoint& ep) const;
```

参数

addr
本机指定网卡ip地址。

port
要绑定的端口。

ep
要绑定的本机地址。

© 2021 yasio.org

8F 56 50
D4 DE 3C

0 : 62 52 FF < 0 59 8D FF 90 8F xxsocket::get_last_errno 83 53 95 8B 78 30
B7 D6 19 EF 01 02

xxsocket::bind_any

7F 5B
D1 9A socket 67 67 4E 61 57 57 30
2C 3A FB 0F 30 40 02

```
int bind_any(bool ipv6) const;
```

53 65
C2 70

ipv6
66 54 7E 5B 67 67 4E 61 IPv6 57 57
2F 28 D1 9A 2C 3A FB 0F 30 40

8F 56 50
D4 DE 3C

0 : 62 52 FF < 0 59 8D FF 90 8F xxsocket::get_last_errno 83 53 95 8B 78 30
B7 D6 19 EF 01 02

xxsocket::listen

5F 59 76 54 67 81 TCP 5B 62 7A 76 63 62 8B 6C 30
00 CB D1 2C 65 EA A2 37 EF 84 E1 4B F7 42 02

```
int listen(int backlog = SOMAXCONN) const;
```

53 65
C2 70

backlog
67 59 76 54 65 30
00 27 D1 2C 70 02

8F 56 50
D4 DE 3C

0 : 62 52 FF < 0 59 8D FF 90 8F xxsocket::get_last_errno 83 53 95 8B 78 30
B7 D6 19 EF 01 02

xxsocket::accept

63 53 4E 4F 5B 62 7A 8F 63 30
A5 D7 00 2A A2 37 EF DE A5 02

```
xxsocket accept() const;
```

© 2021 yasio.org

8F 56 50
D4 DE 3C

54 5B 62 7A 90 4F 76 `xxsocket` 5B 8C 30
8C A2 37 EF 1A E1 84 F9 61 02

xxsocket::accept_n

97 98 58 65 5F 63 53 67 4E 4E 58 62 7A 8F 63 30
5E 3B 5E B9 6F A5 06 2A A2 37 EF DE A5 02

```
int accept_n(socket_native_type& new_sock) const;
```

53 65
C2 70

new_sock

8F 51 53 63 FF 54 5B 62 7A 90 4F 76 5E 5C
93 FA C2 70 0C 8C A2 37 EF 1A E1 64 95 42 socket 53 67 5F 75
C4 15 28

8F 56 50
D4 DE 3C

`0` : 62 52 FF `< 0` 59 8D FF 90 8F `xxsocket::get_last_errno` 83 53 95 8B 78 30
10 9F 6C B7 D6 19 EF 61 02

6C 61
E8 0F

59 67 6B 51 65 8F 56 `0` FF *new_sock* 4F 88 8B 7F 4E 97 96 5B 6A 5F 30
82 9C 64 FD 70 D4 DE 0C 1A AB BE 6E 3A 5E 3B 5E 21 0F 02

8C 75 6B 51 65 4E 52 FF 8B 62 52 8C 75 `xxsocket::set_nonblocking` 5C socket 8B 7F 4E 97 96 58 6A 5F 30
03 28 64 FD 70 4B 4D 0C F7 4B A8 03 28 06 BE 6E 3A 5E 3B 5E 21 0F 02

xxsocket::connect

5E 7A 8F 63 30
FA CB DE A5 02

```
int connect(const char* addr, u_short port);
int connect(const endpoint& ep);
```

53 65
C2 70

addr

8F 7A 4E 67 3A ip 57 30
DC 6B 3B 3A 40 02

port

8F 7A 4E 67 7A 53 30
DC 6B 3B 3A EF E3 02

ep

8F 7A 4E 67 57 57 30
DC 6B 3B 3A 30 40 02

© 2021 yasio.org

8F 56 50
D4 DE 3C

0 : 62 52 FF 6C < 0 59 8D FF 90 8F xxsocket::get_last_errno 83 53 95 8B 78 30
B7 D6 19 EF 61 62

6C 61
E8 0F

TCP: 53 8D TCP 4E 6B 63 62
69 21 E1 4B

UDP: 5E 7A 4 51 7E 7E 58
FA CB 43 C4 D1 9A

xxsocket::connect_n

5E 7A 8F 63 39
FA CB DE A5 62

```
int connect_n(const char* addr, u_short port, const std::chrono::microseconds& wtimeout);
int connect_n(const endpoint& ep, const std::chrono::microseconds& wtimeout);
int connect_n(const endpoint& ep);
```

53 65
C2 70

addr
8F 7A 4E 67
DC 0B 3B 3A ip 57 57 30
30 40 02

port
8F 7A 4E 67 7A 53 39
DC 0B 3B 3A EF E3 02

ep
8F 7A 4E 67 57 57 39
DC 0B 3B 3A 30 40 02

wtimeout
5E 7A 8F 63 76 8D 65 65 95 30
FA CB DE A5 84 85 F6 F6 F4 02

8F 56 50
D4 DE 3C

0 : 62 52 FF 6C < 0 59 8D FF 90 8F xxsocket::get_last_errno 83 53 95 8B 78 39
B7 D6 19 EF 61 62

6C 61
E8 0F

TCP: 53 8D TCP 4E 6B 63 62
77 21 E1 4B

UDP: 5E 7A 4 51 7E 7E 58
FA CB 43 C4 D1 9A

© 2021 yasio.org

xxsocket::connect_n

socket4

```
int disconnect() const;
```

0 : < 0 xxsocket::get_last_errno

SOCK_DGRAM (UDP) socket

xxsocket::send

```
int send(const void* buf, int len, int flags = 0);
```

buf

len

flags

==len : < len xxsocket::get_last_errno

xxsocket::send_n

```
int send_n(const void* buf, int len, const std::chrono::microseconds& wtimeout, int flags =
0);
```

© 2021 yasio.org

53 65
C2 70

buf

89 53 90 65 63 76 8D 59 5B 82 57 57 30
81 D1 01 70 6E 84 77 CB 57 82 30 40 02

len

89 53 90 65 63 76 95 5E 30
81 D1 01 70 6E 84 7F A6 02

wtimeout

53 90 8D 65 65 95 39
D1 01 85 F6 F6 F4 02

flags

53 90 65 63 5E 5C 68 88 30
D1 01 70 6E 95 42 07 B0 02

8F 56 50
D4 DE 3C

==len : 62 52 FF < len 59 8D FF 90 8F xxsocket::get_last_errno 83 53 95 88 78 30
10 9F 6C 31 25 0C 1A C7 B7 D6 19 EF 01 02

xxsocket::recv

4E 51 68 53 96 8F 7A 4E 67 53 90 8F 67 76 65 63 30
CE 85 38 BB 64 DC 6B 3B 3A D1 01 C7 65 64 70 6E 02

```
int recv(void* buf, int len, int flags = 0) const;
```

53 65
C2 70

buf

63 65 65 63 7E 51 53 30
A5 36 70 6E 13 B2 3A 02

len

63 65 65 63 7E 51 53 95 5E 30
A5 36 70 6E 13 B2 3A 7F A6 02

flags

63 65 65 63 5E 5C 68 88 30
A5 36 70 6E 95 42 07 B0 02

8F 56 50
D4 DE 3C

==len : 62 52 FF < len 59 8D FF 90 8F xxsocket::get_last_errno 83 53 95 88 78 30
10 9F 6C 31 25 0C 1A C7 B7 D6 19 EF 01 02

6C 61
E8 0F

68 51 65 60 54 7A 53 8F 56 FF 53 51 4E socket 67 8E 69 54 66 87 98 5E 8A 8F 30
64 F0 70 2F 20 CB 73 D4 DE 0C D6 63 8E 2C AB 2F 20 2F 38 2E 21 0F 02

© 2021 yasio.org

xxsocket::recv_n

[illegible]

```
int recv_n(void* buf, int len, const std::chrono::microseconds& wtimeout, int flags = 0)
const;
```

[illegible]

buf

[illegible]

len

[illegible]

wtimeout

[illegible]

flags

[illegible]

[illegible]

`==len`：[illegible] `< len` [illegible] `xxsocket::get_last_errno` [illegible]

xxsocket::sendto

[illegible] `DGRAM` [illegible]UDP[illegible]

```
int sendto(const void* buf, int len, const endpoint& to, int flags = 0) const;
```

[illegible]

buf

[illegible]

len

[illegible]

to

[illegible]

flags

[illegible]

© 2021 yasio.org

8F 56 50
D4 DE 3C

==len : 62 52 FF 10 9F 6C < len 59 8D FF 90 8F 31 25 0C 1A C7 xxsocket::get_last_errno 83 53 95 8B 78 30 B7 D6 19 EF 01 62

xxsocket::recvfrom

4E 51 68 53 96 8F 7A 4E 67 53 90 8F 67 79 65 63 30 CE 85 38 BB 64 DC 6B 3B 3A D1 01 C7 65 84 70 6E 02

```
int recvfrom(void* buf, int len, endpoint& peer, int flags = 0) const;
```

53 65 C2 70

buf

63 65 65 62 7E 51 53 30 A5 36 70 6E 15 B2 3A 02

len

63 65 65 62 7E 51 53 95 5E 30 A5 36 70 6E 13 B2 3A 7F A6 02

peer

63 65 65 63 67 6E FF 8F 51 53 65 30 A5 36 70 6E 65 96 6C 95 FA C2 70 02

flags

63 65 65 63 5E 5C 68 8B 30 A5 36 70 6E 95 42 07 B0 02

8F 56 50
D4 DE 3C

==len : 62 52 FF 10 9F 6C < len 59 8D FF 90 8F 31 25 0C 1A C7 xxsocket::get_last_errno 83 53 95 8B 78 30 B7 D6 19 EF 01 62

6C 61
E8 0F

6B 51 65 66 54 7A 53 8F 56 FF 53 51 4E 64 F0 70 2F 26 CB 73 D4 DE 6C D6 B3 8E socket 67 8E 66 54 66 2C AB 2F 26 2F 30 02

xxsocket::handle_write_ready

78 5E 49 85 socket 53 51 30 EF 99 02

```
int handle_write_ready(const std::chrono::microseconds& wtimeout) const;
```

53 65 C2 70

wtimeout

7B 5E 80 65 65 95 30 49 85 85 F6 F6 F4 02

© 2021 yasio.org

8F 56 50
D4 DE 3C

0 : 8D 65 FF 6C 1 : 62 52 FF 0C < 0 : 59 8D FF 90 8F 31 25 6C 1A C7 xxsocket::get_last_errno 83 53 95 8B 78 30 B7 D6 19 EF 61 02

6C 61
E8 0F

90 5E 5F 51 68 53 09 7F 51 53 6C 6E 76 69 51 4E FF 6B 51 65 4F 7A 53 8F 56 39 1A 38 55 85 58 D1 01 15 B2 3A A1 E1 84 C9 B5 0B 0C 64 FD 70 1A CB 73 D4 DE 02

xxsocket::handle_read_ready

7B 5E socket 53 8B 39 EF FB 02

```
int handle_read_ready(const std::chrono::microseconds& wtimeout) const;
```

53 65
C2 70

wtimeout
7B 5E 8D 65 65 95 39 49 85 85 F6 F6 F4 02

8F 56 50
D4 DE 3C

0 : 8D 65 FF 6C 1 : 62 52 FF 0C < 0 : 59 8D FF 90 8F 31 25 6C 1A C7 xxsocket::get_last_errno 83 53 95 8B 78 30 B7 D6 19 EF 61 02

xxsocket::local_endpoint

83 53 4 51 7E 90 4F 76 67 57 57 39 B7 D6 43 C4 1A E1 84 2C 30 57 40 02

```
endpoint local_endpoint() const;
```

8F 56 50
D4 DE 3C

8F 56 67 57 57 39
D4 DE 2C 30 30 40 02

6C 61
E8 0F

59 67 6C 8C 75 8F 82 9C A1 69 83 28 C7 xxsocket::connect 62 80 TCP 8E 3 63 6B 21 63 E1 62 4B 67 2A 5B 8C 62 10 FF 90 0C 6C A3 4E 8F 48 59 D4 DE 76 57 84 57 60 30 2F 0.0.0.0

xxsocket::peer_endpoint

83 53 4 51 7E 90 4F 76 5B 7A 57 39 B7 D6 43 C4 1A E1 84 F9 EF 30 40 02

© 2021 yasio.org

```
endpoint peer_endpoint() const;
```

8F 56 50
D4 DE 3C

8F 56 67 57 57 57 39
D4 DE 2C 50 30 40 02

6C 61
E8 0F

59 67 6C A1 67 8C 75 8F `xxsocket::connect` 62 80 TCP 8E 63 3 6B 63 62 67 5B 62 FF 90 4E 8F 56 76 57 57 60
82 9C A1 69 03 28 C7 21 E1 4B 2A 8C 10 6C A3 48 D4 DE 84 30 40 2F `0.0.0.0`

xxsocket::set_keepalive

8B 7E TCP 5E 5C 53 8B 76 5F 8D 53 65 30
BE 6E 05 42 4F AE 84 C3 F3 C2 70 02

```
int set_keepalive(int flag = 1, int idle = 7200, int interval = 75, int probes = 10);
```

53 65
C2 70

flag

`1`：5F 54 5E 5C 53 8B 5E 8D FF `0`：51 95 30
00 2F 95 42 4F AE C3 F3 6C 73 ED 02

idle

5F 5E 75 5C 6C 67 4E 4F 6D 60 4E 4E 54 FF 54 52 5E 5C 53 8B 5E 8D 63 6D 76 67 59 8D 65 95 FF 53 4F FF 79 FF 30
53 94 28 42 A1 69 FB 55 88 6F A4 92 0E 6C 2F A8 95 42 4F AE C3 F3 A2 4B 84 60 27 85 F6 F4 6C 55 4D 68 D2 69 02

interval

5F 6C 67 65 52 5F 8D 56 5E 65 FF 91 59 53 90 5F 8D 63 6D 62 65 95 95 96 FF 53 4F FF 79 FF 30
53 A1 69 36 30 C3 F3 DE 94 F6 6C CD 60 D1 01 C3 F3 A2 4B A5 F6 F4 F4 94 6C 55 4D 68 D2 69 02

probes

5F 6C 67 65 52 5F 8D 56 5E 65 FF 67 59 63 6D 6B 65 FF 8D 8F 63 6D 6B 65 54 FF 4F 89 53 5E 75 5C 8F 63 65 5F 30
53 A1 69 36 30 C3 F3 DE 94 F6 6C 60 27 A2 4B 21 70 6C 85 C7 A2 4B 21 70 0E 6C 1A E8 D1 94 28 42 DE A5 AD 00 02

8F 56 50
D4 DE 3C

`0`：62 52 FF `< 0` 59 8D FF 90 8F `xxsocket::get_last_errno` 83 53 95 8B 78 30
31 25 6C 1A C7 B7 D8 19 EF 01 02

79 4F
3A 8B

```
// xxsocket-keepalive.cpp
#include "yasio/xxsocket.hpp"
using namespace yasio;
using namespace inet;

int main(){
    xxsocket client;
    if(0 == client.pconnect("192.168.1.19", 80)) {
```

© 2021 yasio.org

```
        client.set_keepalive(1, 5, 10, 2);
    }
    return 0;
}
```

xxsocket::reuse_address

[illegible]socket[illegible]

```
void reuse_address(bool reuse);
```

[illegible]

reuse
[illegible]

[illegible]

[illegible]

xxsocket::exclusive_address

[illegible]

```
void exclusive_address(bool exclusive);
```

[illegible]

exclusive
`true`:[illegible] `false`:[illegible]

[illegible]

[illegible] `winsock` [illegible]

xxsocket::set_optval

[illegible]socket[illegible]

```
template <typename _Ty> int set_optval(int level, int optname, const _Ty& optval);
```

© 2021 yasio.org

level
socket

optname

optval

`0` `< 0` `xxsocket::get_last_errno`

bsd socket `setsockopt`

xxsocket::get_optval

socket

```
template <typename _Ty> _Ty get_optval(int level, int optname) const
```

level
socket

optname

bsd socket `getsockopt`

© 2021 yasio.org

xxsocket::select

socket

```
int select(fd_set* readfds, fd_set* writefds, fd_set* exceptfds, const
std::chrono::microseconds& wtimeout)
```

readfds

writefds

exceptfds

wtimeout

`0` : `> 0` : `< 0` : `xxsocket::get_last_errno`

xxsocket::shutdown

TCP

```
int shutdown(int how = SD_BOTH) const;
```

how

- `SD_SEND` :
- `SD_RECEIVE` :
- `SD_BOTH` :

© 2021 yasio.org

8F 56 50
D4 DE 3C

0 : 62 52 FF < 0 : 59 8D FF 90 8F xxsocket::get_last_errno 83 53 95 8B 78 30
10 9F 6C 31 25 6C 1A C7 B7 D6 19 EF 01 02

xxsocket::close

51 95
73 ED socket FF 91 65 7C 7E 8D 6E 39
6C CA 3E FB DF 44 96 02

```
void close(int shut_how = SD_BOTH);
```

53 65
C2 70

shut_how
51 95 90 99 7C 57 FF 53 4F 4E 4E 67 4E 50
73 ED 1A 53 78 80 6E EF E5 20 E5 0B 9A 3E 3C

- SD_SEND : 53 90 90 99
  D1 01 1A 53
- SD_RECEIVE : 63 53 90 99
  A5 D7 1A 53
- SD_BOTH : 51 90 51 95
  68 E8 73 ED
- SD_NONE : 51 90 51 95
  68 E8 73 ED

8F 56 50
D4 DE 3C

0 : 62 52 FF < 0 : 59 8D FF 90 8F xxsocket::get_last_errno 83 53 95 8B 78 30
10 9F 6C 31 25 6C 1A C7 B7 D6 19 EF 01 02

6C 61
E8 0F

59 67 *shut_how* != SD_NONE FF 68 51 65 4F 51 8C 75 shutdown FF 51 8C 75 5E 5C close 30
82 9C 6C 64 FD 70 1A 48 03 28 6C 8D 03 28 95 42 02

xxsocket::tcp_rtt

83 53 TCP 8F 63 76 RTT 30
B7 D6 DE A5 84 02

```
uint32_t tcp_rtt() const;
```

8F 56 50
D4 DE 3C

8F 56 TCP 76 RTT 65 95 FF 53 4F: 5E 79 30
D4 DE 84 F6 F4 0C 55 4D AE D2 02

© 2021 yasio.org

xxsocket::get_last_errno

socket

```
  static int get_last_errno();
```

`0` > 0 xxsocket::strerror

xxsocket::set_last_errno

socket

```
  static void set_last_errno(int error);
```

error

xxsocket::not_send_error

socket

```
  static bool not_send_error(int error);
```

error

© 2021 yasio.org

8F 56 50
D4 DE 3C

`true` : socket 69 5E FF 63 38 0C `false` : socket 77 60 5D 7E 53 72 95 8B FF 5E 5F 51 95 B6 01 F2 CF D1 1F 19 EF 0C 94 53 73 ED socket 7E 6B 90 8B 30 C8 62 1A AF 02

6C 61
E8 0F

4E 5F 53 90 64 4F 8F 56 50 C5 53 01 01 CD 5C 84 DE 3C `< 0` 65 FF 8C 75 6B 51 65 30 F6 6C 83 28 64 FD 78 02

xxsocket::not_recv_error

52 65 69 54 69 63 65 65 24 AD 2F 26 2F A5 36 F6 socket 51 73 65 6C 7E 7E 76 95 8B 30 FA 60 E0 D5 E7 ED 84 19 EF 02

```
static bool not_recv_error(int error);
```

53 65
C2 70

error
95 8B 78 30
19 EF 01 02

8F 56 50
D4 DE 3C

`true` : socket 69 5E FF 63 38 0C `false` : socket 77 60 5D 7E 53 72 95 8B FF 5E 5F 51 95 B6 01 F2 CF D1 1F 19 EF 0C 94 53 73 ED socket 7E 6B 90 8B 30 C8 62 1A AF 02

6C 61
E8 0F

4E 5F 63 65 64 4F 8F 56 50 C5 53 A5 36 CD 5C 84 DE 3C `< 0` 65 FF 8C 75 6B 51 65 30 F6 6C 83 28 64 FD 78 02

xxsocket::strerror

5C 95 8B 78 8F 63 4E 5B 78 4E 30
06 19 EF 01 6C 62 3A 57 26 32 02

```
static const char* strerror(int error);
```

53 65
C2 70

error
95 8B 78 30
19 EF 01 02

8F 56 50
D4 DE 3C

95 8B 4F 60 76 5B 7B 4E 30
19 EF E1 6F 84 57 26 32 02

© 2021 yasio.org

xxsocket::gai_strerror

5C 06 `getaddrinfo` 95 8B 78 8F 63 4E 58 7B 4E 30 19 EF 01 6C 62 3A 57 28 32 02

```
static const char* gai_strerror(int error);
```

53 65 C2 70

error

95 8B 78 30 19 EF 01 02

8F 56 50 D4 DE 3C

95 8B 4F 60 76 58 7B 4E 30 19 EF E1 6F 84 57 28 32 02

xxsocket::resolve

89 67 57 54 53 54 76 62 67 57 57 30 E3 96 DF 6D 05 2B 84 40 09 30 40 02

```
int resolve(std::vector<endpoint>& endpoints, const char* hostname, unsigned short port = 0,
int socktype = SOCK_STREAM);
```

53 65 C2 70

endpoints

8F 51 53 63 39 93 FA C2 70 02

hostname

57 54 30 DF 6D 02

port

7A 53 30 EF E3 02

socktype

socket 7C 57 39 7B 88 02

8F 56 50 D4 DE 3C

`0`：65 95 8B FF E0 19 EF 0C `> 0` 90 8F 1A C7 `xxsocket::strerror` 8F 63 4E 8B 7E 95 8B 4F 60 30 6C 62 3A E6 C6 19 EF E1 6F 02

© 2021 yasio.org

xxsocket::resolve_v4

IPv4

```
int resolve_v4(std::vector<endpoint>& endpoints, const char* hostname, unsigned short port =
0, int socktype = SOCK_STREAM);
```

endpoints

hostname

port

socktype

socket

`0` `> 0` `xxsocket::strerror`

xxsocket::resolve_v6

IPv6

```
int resolve_v6(std::vector<endpoint>& endpoints, const char* hostname, unsigned short port =
0, int socktype = SOCK_STREAM);
```

endpoints

hostname

port

© 2021 yasio.org

socktype

socket

0 : > 0 xxsocket::strerror

xxsocket::resolve_v4to6

IPv4IPv6 V4MAPPED

```
int resolve_v4to6(std::vector<endpoint>& endpoints, const char* hostname, unsigned short port
= 0, int socktype = SOCK_STREAM);
```

endpoints

hostname

port

socktype

socket

0 : > 0 xxsocket::strerror

xxsocket::resolve_tov6

IPv4IPv6 V4MAPPED

```
int resolve_tov6(std::vector<endpoint>& endpoints, const char* hostname, unsigned short port =
0, int socktype = SOCK_STREAM);
```

© 2021 yasio.org

endpoints

hostname

port

socktype

socket

`0` : `> 0` `xxsocket::strerror`

xxsocket::getipsv

IP

```
static int getipsv();
```

- `ipsv_ipv4` : IPv4
- `ipsv_ipv6` : IPv6
- `ipsv_dual_stack` : IPv4IPv6

IPv4

`wifi` wifi

wifiIPv4: https://github.com/halx99/yasio/issues/130

```
// xxsocket-ipsv.cpp
#include <vector>
#include "yasio/xxsocket.hpp"
using namespace yasio;
using namespace yasio::inet;
```

© 2021 yasio.org

```
int main(){
    const char* host = "github.com";
    std::vector<ip::endpoint> eps;
    int flags = xxsocket::get_ipsv();
    if(flags & ipsv_ipv4) {
        xxsocket::resolve_v4(eps, host, 80);
    }
    else if(flags & ipsv_ipv6) {
        xxsocket::resolve_tov6(eps, host, 80);
    }
    else {
        std::cerr << "Local network not available!\n";
    }
    return 0;
}
```

xxsocket::traverse_local_address

67 4E 67 67 57 57 39 9A 3F 2C 3A 30 40 02

```
static void traverse_local_address(std::function<bool(const ip::endpoint&)> handler);
```

53 65 C2 70

handler

67 4E 57 57 56 8C 39 9A 3F 30 40 DE 63 02

79 4F 3A 8B

```
// xxsocket-traverse.cpp
#include <vector>
#include "yasio/xxsocket.hpp"
using namespace yasio;
using namespace yasio::inet;
int main(){
    int flags = 0;
    xxsocket::traverse_local_address([&](const ip::endpoint& ep) -> bool {
        switch (ep.af())
        {
          case AF_INET:
            flags |= ipsv_ipv4;
            break;
          case AF_INET6:
            flags |= ipsv_ipv6;
            break;
        }
        return (flags == ipsv_dual_stack);
    });
    YASIO_LOG("Supported ip stack flags=%d", flags);
```

© 2021 yasio.org

```
    return flags;
}
```

8B53
F7C2

io_service Class

© 2021 yasio.org

© 2021 yasio.org

io_service options

io_service

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*

© 2021 yasio.org

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),

© 2021 yasio.org

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

© 2021 yasio.org

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

io_service Class

© 2021 yasio.org

© 2021 yasio.org

51 4E 5E 75 76 D6 38 28 API

54 54 7A 95 7D 0D 7A F4

```
namespace yasio {}
```

51 51 65 6C 6C 71 B9 D5

Name	Description
yasio::host_to_network	4E 67 59 82 5E 8F 7F 7E 5B 82 5E 3B 9A 57 82 8F 6C 51 DC 57 82 8F
yasio::network_to_host	7F 7E 5B 82 5E 8F 4E 67 5B 82 5E 51 DC 57 B2 8F 6C 3B 9A 57 82 8F
yasio::xhighp_clock	83 53 7E 79 7E 65 95 62 B7 D6 83 D2 A7 F6 F4 33
yasio::highp_clock	83 53 5E 79 7E 65 95 62 B7 D6 AE D2 A7 F6 F4 33
yasio::clock	83 53 6B 79 7E 65 95 62 B7 D6 EB D2 A7 F6 F4 33
yasio::set_thread_name	8B 7E 8C 75 80 7E 7A 54 8E 6E 83 28 85 6F 0B 0D
yasio::basic_strfmt	68 5E 53 59 78 45 3C 0F 16 57 26 32

yasio::host_to_network

4E 67 59 82 5E 8F 8C 51 DC 57 82 8F 39 02

59 65 4E 34 87 F6

yasio/detail/endian_portable.hpp

```
template <typename _Ty>
inline _Ty host_to_network(_Ty value);
inline int host_to_network(int value, int size);
```

© 2021 yasio.org

53 65
C2 70

value

89 8F 63 76 65 50 FF value 7C 57 _Ty 53 4E 69 4F 61 1~8 5B 82 65 50 7C 57 30
81 6C 62 84 70 3C 6C 7B 8B EF E5 2F FB 0F 57 82 70 3C 7B 88 02

size

89 8F 63 76 65 50 67 65 5B 82 65 FF 53 80 66 1~4 5B 82 30
81 6C 62 84 70 3C 69 48 57 82 70 6C EA FD 2F 57 82 02

8F 56 50
D4 DF 3C

7F 7E 58 82 5E 65 50 30
51 DC 57 82 8F 70 3C 02

79 4F
3A 8B

```
#include <stdio.h>
#include <inttypes.h>
#include "yasio/detail/endian_portable.hpp"
int main(){
    uint16_t v1 = 0x1122;
    uint32_t v2 = 0x11223344;
    uint64_t v3 = 0x1122334455667788;
    // output will be: net.v1=2211, net.v2=44332211, net.v3=8877665544332211
    printf("net.v1=%04" PRIx16 ", net.v2=%08" PRIx32 ", net.v3=%016" PRIx64 "\n",
        yasio::host_to_network(v1),
        yasio::host_to_network(v2),
        yasio::host_to_network(v3));
    return 0;
}
```

yasio::network_to_host

7F 7E 58 82 5E 8F 4E 67 5B 82 5E 30
51 DC 57 82 8F 6C 3B 3A 57 82 8F 02

59 65 4E
34 87 F6

`yasio/detail/endian_portable.hpp`

```
template <typename _Ty>
inline _Ty network_to_host(_Ty value);
inline int network_to_host(int value, int size);
```

53 65
C2 70

value

89 8F 63 76 65 50 FF value 7C 57 _Ty 53 4E 69 4F 61 1~8 5B 82 65 50 7C 57 30
81 6C 62 84 70 3C 6C 7B 8B EF E5 2F FB 0F 57 82 70 3C 7B 88 02

© 2021 yasio.org

size

1~4

```
#include <stdio.h>
#include <inttypes.h>
#include "yasio/detail/endian_portable.hpp"
int main(){
    uint16_t v1 = 0x2211;
    uint32_t v2 = 0x44332211;
    uint64_t v3 = 0x8877665533442211;
    // output will be: net.v1=1122, net.v2=11223344, net.v3=1122334455667788
    printf("host.v1=%04" PRIx16 ", host.v2=%08" PRIx32 ", host.v3=%016" PRIx64 "\n",
        yasio::network_to_host(v1),
        yasio::network_to_host(v2),
        yasio::network_to_host(v3));
    return 0;
}
```

yasio::xhighp_clock

`yasio/detail/utils.hpp`

```
template <typename _Ty = steady_clock_t>
inline highp_time_t xhighp_clock();
```

_Ty

- yasio::steady_clock_t:
- yasio::system_clock_t: UTC

© 2021 yasio.org

yasio::highp_clock

yasio/detail/utils.hpp

```
template <typename _Ty = steady_clock_t>
inline highp_time_t highp_clock();
```

_Ty

- `yasio::steady_clock_t`
- `yasio::system_clock_t`UTC

yasio::clock

yasio/detail/utils.hpp

```
template <typename _Ty = steady_clock_t>
inline highp_time_t clock();
```

_Ty

- `yasio::steady_clock_t`

© 2021 yasio.org

- `yasio::system_clock_t` : 8F 56 50 D4 DE 7C 7E DF UTC 65 F6 95 F4 62 33

8F 56 50
D4 DE 3C

6B 79 7E 65 95 62 39
EB D2 A7 F6 F4 33 02

yasio::set_thread_name

8B 7E 8C 75 89 7E 7A 54 30
BE 6E 03 28 05 BF 0B 0D 02

59 65 4E
34 87 F6

`yasio/detail/thread_name.hpp`

```
inline void set_thread_name(const char* name);
```

53 65
C2 70

name
97 89 8B 7F 7E 7A 54 79
00 81 BE 6E BF 0B 0D F0

6C 61
E8 0F

6B 51 65 75 4E 8B 65 59 7E 7A 7A 5E 97 5E 67 75 30
64 FD 70 28 8E CA AD 1A BF 0B 0B 8F 5E 38 09 28 02

yasio::basic_strfmt

51 65 6A 67 FF 68 5E 53 5B 7B 4E 62 80 5B 5B 7B 4E 30
FD 70 21 7F 6C 3C 0F 16 57 26 32 16 05 BD 57 26 32 02

59 65 4E
34 87 F6

`yasio/detail/strfmt.hpp`

```
template <class _Elem, class _Traits = std::char_traits<_Elem>,
          class _Alloc = std::allocator<_Elem>>
inline std::basic_string<_Elem, _Traits, _Alloc> basic_strfmt(size_t n, const _Elem*
format, ...);
```

© 2021 yasio.org

53 65 C2 70

n

52 59 1D CB buffer 59 5C 27 0F

format

68 5F 3C 0F 58 57 7B 26 4E 32 FF 0C 54 8C 68 07 51 C6 5E 93 `printf` 76 54 F8 6C

8F 56 50 D4 DE 3C

8F 56 68 5F 53 54 76 D4 DE 3C 0F 16 6E 84 `std::string` 7C 57 7B 8B 58 57 7B 26 4E 32 30 62

6C 61 E8 0F

5B 96 4F 75 FF 53 76 63 4F 75 5D 7E 58 4E 59 76 52 54 9E 45 7F 28 6C EF F4 A5 7F 28 F2 CF 9A 49 70 64 26 60

- `yasio::strfmt`：68 5F 53 58 7B 4E 32 0F 16 57 26 32
- `yasio::wcsfmt`：68 5F 53 58 58 7B 4E 32 0F 16 60 57 26 52

79 4F 3A 8B

```
#include "yasio/detail/strfmt.hpp"
int main() {
    std::string str1 = yasio::strfmt(64, "My age is %d", 19);
    std::wstring str2 = yasio::wcsfmt(64, L"My age is %d", 19);
    return 0;
}
```

© 2021 yasio.org

yasio 宏定义

以下宏定义可以控制 `yasio` 库的某些行为，可以在 yasio/detail/config.hpp 定义或者在编译器预处理器定义

Name	Description
YASIO_HAVE_KCP	是否启用kcp传输支持，需要kcp已经在软件编译系统中，默认关闭。
YASIO_HEADER_ONLY	是否以仅头文件的方式使用yasio核心组件，默认关闭。
YASIO_SSL_BACKEND	选择SSL库以支持SSL客户端，需要软件编译系统包含OpenSSL/MbedTLS库，3.36.0新增(同时移除YASIO_HAVE_SSL)，此宏只能取值 `1` (使用OpenSSL) 或者 `2` (使用mbedtls)。
YASIO_ENABLE_UDS	是否启用unix domain socket支持，目前仅类unix系统和win10 RS5+支持，默认关闭。
YASIO_HAVE_CARES	是否启用c-ares异步域名解析库， 当编译系统包含c-ares时可启用，有效避免每次解析域名都新开线程。 yasio有DNS缓存机制，超时时间默认10分钟， 因此无c-ares也不会造成太大的性能损耗。
YASIO_VERBOSE_LOG	是否打印详细日志，默认关闭。
YASIO_NT_COMPAT_GAI	是否启用Windows XP系统下使用 `getaddrinfo` API支持。
YASIO_USE_SPSC_QUEUE	是否使用SPSC(单生产者单消费者)队列， 仅当只有一个线程调用io_service::write时方可启用，默认关闭。
YASIO_USE_SHARED_PACKET	是否使用 `std::shared_ptr` 包装网络包，使其能在多线程之间共享，默认关闭。
YASIO_HAVE_HALF_FLOAT	是否启用半精度浮点数支持，依赖 half.hpp。
YASIO_DISABLE_OBJECT_POOL	是否禁用对象池的使用，默认启用。
YASIO_DISABLE_CONCURRENT_SINGLETON	是否禁用并发单例类模板。

© 2021 yasio.org

© 2021 yasio.org

yasio 粘包处理

yasio的粘包处理不仅针对TCP，对于UDP，如果发送端有组包发送机制，也是以相同的方式处理。整体来讲有两种方式:

- 通过io_service选项 YOPT_C_LFBFD_PARAMS 设置信道参数。
- 通过io_service选项 `YOPT_C_LFBFD_FN` 设置自定义包长度解码函数 decode_len_fn_t 。

> **注意**
>
> 自定义解码包长度函数实现时，当从字节流中读取int值时，一定不要使用指针强转，否则可能触发ARM芯片字节对齐问题 `SIGBUS` 异常闪退。可以参考内置解码包长度函数实现 `io_channel::__builtin_decode_len` 。

YOPT_C_LFBFD_PARAMS

设置信道拆包参数。

参数

max_frame_length
最大包长度，超过将视为异常包。

length_field_offset
包长度字段相对于消息包数据首字节偏移。

length_field_length
包长度字段大小，支持1~4字节整数(`uint8_t,uint16_t,uint24_t,int32_t`)。

length_adjustment
包长度调整值。通常应用层二进制协议都会设计消息头和消息体，当长度字段值包含消息头时，则此值为 `0` ，否则为 `消息头长度` 。

注意

消息包长度字段必须使用网络字节序编码。请查看内置解码包长度实现:

```
int io_channel::__builtin_decode_len(void* d, int n)
{
  int loffset = uparams_.length_field_offset;
  int lsize   = uparams_.length_field_length;
  if (loffset >= 0)
  {
    int len = 0;
    if (n >= (loffset + lsize))
```

© 2021 yasio.org

```
    {
      ::memcpy(&len, (uint8_t*)d + loffset, lsize);
      len = yasio::network_to_host(len, lsize);
      len += uparams_.length_adjustment;
      if (len > uparams_.max_frame_length)
        len = -1;
    }
    return len;
  }
  return n;
}
```

decode_len_fn_t

4F 90 89 78 6D 60 53 95 5E 51 65 53 57 39
E1 53 E3 01 88 6F 05 7F A6 FD 70 9F 8B 02

```
typedef std::function<int(void* d, int n)> decode_len_fn_t;
```

53 65
C2 70

d

5F 89 53 65 63 99 58 82 57 57 39
85 E3 05 70 62 96 57 82 36 40 02

n

5F 89 53 65 63 95 5E 30
85 E3 05 70 6E 7F A6 02

8F 56 50
D4 DE 3C

8F 56 6D 60 53 65 63 58 96 95 5E 30
D4 DE 88 6F 05 70 6E 9E 45 7F A6 02

- `> 0`：89 78 53 95 5E 62 52 30
E3 01 65 7F A6 10 9F 02
- `== 0`：63 65 65 63 4E 8D 4E 89 78 6D 60 53 6B 96 95 5E FF yasio 5E 5C 4F 7E 7E 63 65 65 63 FF 4E 65 65 52 65 65 63 FF 4F 51 60 8C 75 68 51 65 30
95 42 1A E7 ED A3 36 70 62 EC 66 E6 36 30 80 78 6E 6F 1A 8D 21 83 28 64 FD 70 02
- `< 0`：89 78 53 95 5E 5F 5E FF 4F 89 53 5E 52 4F 8F 4F 8B 65 5E 39
E3 01 65 7F A6 02 38 6C 1A E6 D1 53 4D 20 95 1A DD AD 00 02

6C 61
E8 0F

89 78 6D 60 53 95 5E 51 65 5E 98 66 7E 7A 5B 51 76 FF 4F 59 5B 4E Lua 52 4E 63 63 8B 7E 81 5B 4E 89 78 6D 60 53 92 5E 51 65 30
E3 01 88 6F 05 7F A6 FD 70 C5 7B 2F BF 6B 89 68 84 0C 8B 82 F9 8E 19 0D 2F 01 BE 6E EA 9A 49 E3 01 88 6F 05 7F A6 FD 70 02

© 2021 yasio.org

yasio Interop

Unity C# yasioC: https://github.com/yasio/yasio/blob/master/yasio/bindings/yasio_ni.cpp

OpenNSM2

Unity yasio-3.37.1NetworkServiceManagerunity: https://github.com/yasio/OpenNSM2

© 2021 yasio.org

© 2021 yasio.org

FAQ

重点问题解答

yasio是否依赖其他网络库?

yasio的核心代码默认不依赖任何第三方库，从作者从业踏入通信行业开始就琢磨着编写一个轻量而好用的通用网络库，xxsocket就是yasio的起源，只有异步消息发送的软中断器提取于boost.asio。

为什么使用yasio?

1. 开源社区已知比较有名的网络库有asio,libevent,libev,libuv，他们提供的都是非常基础的非阻塞多路io复用模型，并且，各平台底层会使用iocp,kqueue,epoll,select等模型，拿来做客户端网络，如连接管理，TCP粘包处理等都需要程序员自己处理。
2. yasio将连接管理，TCP拆包都封装到了底层。
3. yasio将TCP,UDP,KCP统一抽象成Transport更加方便使用。
4. yasio更轻量级，所有平台均使用select模型。

yasio是否支持非阻塞域名解析?

支持，但需要依赖c-ares库，这个库是完全实现DNS协议，并可以很好的和现有的select模型结合使用，域名解析无需新开线程。如果不开启c-ares，yasio内部会为每次域名解析开线程。但会默认缓存10分钟，所以也不用担心开线程太频繁。

yasio是否支持SSL/TLS传输?

支持，但需要依赖OpenSSL或者MbedTLS。

如何从 io_event 中设置和获取 userdata

请参考：https://github.com/yasio/ftp_server/blob/master/ftp_server.cpp#L98

yasio是否处理 SIGPIPE 信号？

从 `3.30` 版本开始已处理。

如果不处理，实测过iOS在后台锁屏返回时会触发SIGPIPE信号直接导致APP闪退，处理后，仅会触发TCP连接断开，错误码为32，错误信息: `Broken pipe`

详见: https://github.com/yasio/yasio/issues/170

© 2021 yasio.org

iOS10161ec=65, detail:No route to host

- :
  - APPios14.1+ `Local Network Permission`
  - 10161 `SNMP via TLS`
- : APP 49152~65535
- : https://www.speedguide.net/port.php

macOS UDP40 `Message too long`

- : macOSUDP `9126`
- : socketUDP:
  - xxsocket: `sock_udp.set_optval(SOL_SOCKET, SO_SNDBUF, (int)65535);`
  - io_service: https://github.com/yasio/yasio/blob/master/tests/speed/main.cpp#L223

io_service schedule

std::thread

: https://github.com/yasio/yasio/issues/244

xxsocket _n_i

_nnonblock_iinternal()

YOPT_TCP_KEEPALIVE

: https://github.com/yasio/yasio/issues/117

© 2021 yasio.org

Lua

- c++11:
  - kaguya c++ lua_newuserdata, placement new xcode clang release
  - bing.com `LUAI_USER_ALIGNMENT_T=max_align_t`
  - , max_align_t xcode clang long double 16
  - : http://lua-users.org/lists/lua-l/2019-07/msg00197.html
- c++14:
  - sol2 sol2 kaguya sol2 lua_newuserdata clang release
- c++17:
  - sol2 xcode ios11 C++17 `new` `Aligned allocation/deallocation` -faligned-allocation
  - ios10 stl shared_mutex(yasio Apple pthread)
-
  - Lua double `max_align_t` C11 https://en.cppreference.com/w/c/types/max_align_t Lua ANSI C89: `LUAI_USER_ALIGNMENT_T`
  - C++11 `std::max_align_t` : https://en.cppreference.com/w/cpp/types/max_align_t

: **max_align_t llvm __stddef_max_align_t.h**

Can't load xxx.bundle on macOS?

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

xxsocket resolve socktype

- winsock
- O IP