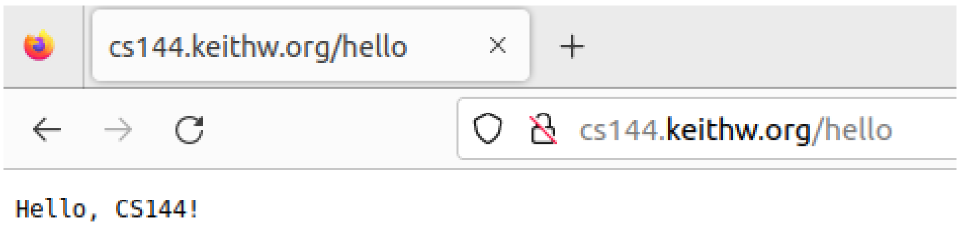
check 0

2. Networking by hands

2.1 Fetch a Web Page

1.在浏览器中访问，结果如下：

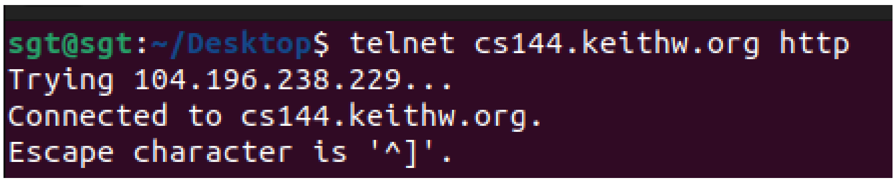


```
Hello, CS144!
```

2.

a.使用终端访问

```
telnet cs144.keithw.org http
```



```
sgt@sgt:~/Desktop$ telnet cs144.keithw.org http
Trying 104.196.238.229...
Connected to cs144.keithw.org.
Escape character is '^]'.
```

b.输入 GET /Hello HTTP/1.1

c.输入Host:cs144.keithw.org

d.输入Connect：close

```
telnet cs144.keithw.org http                #input
Trying 104.196.238.229...
Connected to cs144.keithw.org.
Escape character is '^]'.
GET /lab0/misaka HTTP/1.1                   #input
Host: cs144.keithw.org                      #input
Connection: close                           #input

HTTP/1.1 200 OK
Date: Tue, 05 Dec 2023 01:45:24 GMT
```

```
Server: Apache
X-You-Said-Your-SunetID-Was: misaka
X-Your-Code-Is: 918683
Content-length: 110
Vary: Accept-Encoding
Connection: close
Content-Type: text/plain

Hello! You told us that your SUNet ID was "misaka". Please see the HTTP headers (above) for your secret code.
Connection closed by foreign host.
```

3.访问知乎:

输入以下命令:

```
telnet zhihu.com http
GET /people/deng-feng-lai-62-55 HTTP/1.1
Host:zhihu.com
Connection:close
```

结果如下:

```
telnet zhihu.com http
Trying 103.41.167.234...
Connected to zhihu.com.
Escape character is '^]'.
GET /people/deng-feng-lai-62-55 HTTP/1.1
Host:zhihu.com
Connection:close

HTTP/1.1 301 Moved Permanently
Server: CLOUD ELB 1.0.0
Date: Tue, 05 Dec 2023 02:33:34 GMT
Content-Type: text/html
Content-Length: 182
Connection: close
Location: https://www.zhihu.com/people/deng-feng-lai-62-55
X-Backend-Response: 0.000
Vary: Accept-Encoding
Referrer-Policy: no-referrer-when-downgrade
X-SecNG-Response: 0.0010001659393311
x-lb-timing: 51.921
x-idc-id: 2
Set-Cookie: KLBRSID=4843ceb2c0de43091e0ff7c22eadca8c|1701743614|1701743614; Path=/
```

```

<html>
<head><title>301 Moved Permanently</title></head>
<body bgcolor="white">
<center><h1>301 Moved Permanently</h1></center>
<hr><center>openresty</center>
</body>
</html>
Connection closed by foreign host.
```

2.2 Send yourself an email

1.telnet 148.163.153.234 smtp

```
telnet 148.163.153.234 smtp
Trying 148.163.153.234...
Connected to 148.163.153.234.
Escape character is '^]'.
220 mx0b-00000d03.pphosted.com ESMTP mfa-m0214089
```

2.输入 Helo mycomputer.stanford.edu

```
Helo mycomputer.stanford.edu
250 mx0b-00000d03.pphosted.com Hello [123.127.218.123], pleased to meet you
```

3.输入MAIL FROM：964642078@qq.com，看是谁再发邮件。

```
MAIL FROM:964642078@qq.com
250 2.1.0 Sender ok
```

4.输入 RCPT TO:964642078@qq.com，给自己发邮件

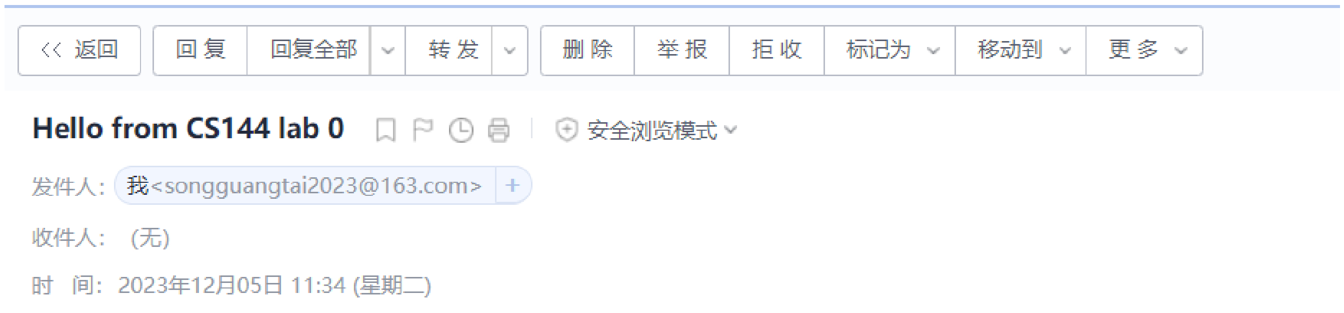
```
RCPT TO:964642078@qq.com
550 5.7.1 Relaying denied
```

会被拒绝，我无法通过stanford的smtp服务器向我自己发邮件。

所以尝试163邮箱的服务器，步骤类似。

```
telnet smtp.163.com 25
Trying 123.126.97.113...
Connected to smtp.163.com.
Escape character is '^]'.
220 163.com Anti-spam GT for Coremail System (163com[20141201])
HElO 163.com
250 OK
auth login
334 dXNlcm5hbWU6
c29uZ2d1YW5ndGFpMjAyMw==                   #163要开启SMTP授权（base64加密用户名）
334 UGFzc3dvcmQ6
WUtES1dSTEZSWlhOQUpSVA==                   #163要开启SMTP授权（base64加密授权码）
235 Authentication successful
MAil FROM:songguangtai2023@163.com
500 Error: bad syntax
MAIL FROM:songguangtai@163.com
500 Error: bad syntax
mail from:<songguangtai2023@163.com>
250 Mail OK
PCPT TO:<songguangtai2023@163.com>
502 Error: command not implemented
RCPT TO:<songguangtai2023@163.com>
250 Mail OK
DATA
354 End data with <CR><LF>.<CR><LF>
subject:Hello from CS144 lab 0

.
250 Mail OK queued as zwqz-smtp-mta-g0-1,_____wAnN9X8mG5lOKePCg--.61839S4 1701747262
```



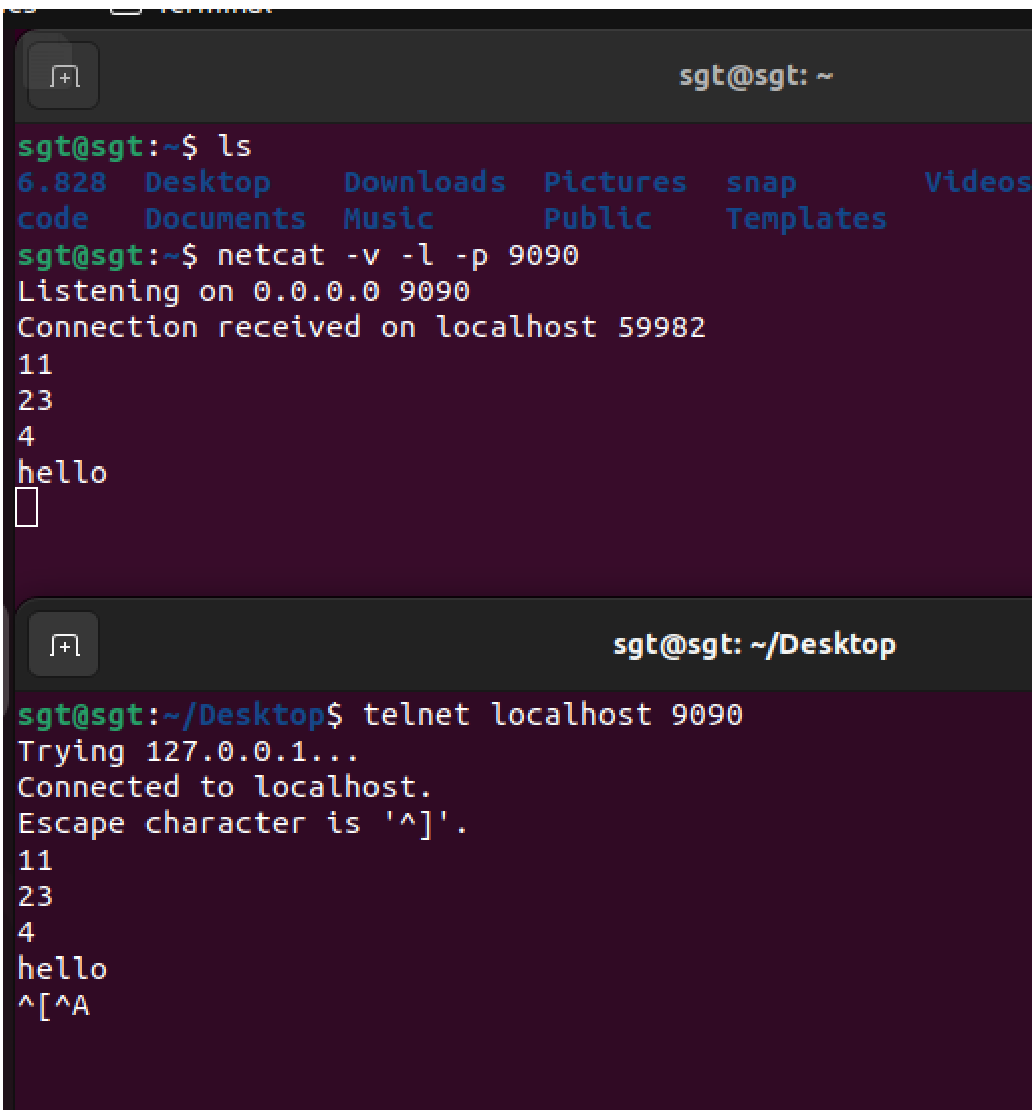
Hello from CS144 lab 0

发件人：我<songguangtai2023@163.com>

收件人：（无）

时　间：2023年12月05日 11:34 (星期二)

2.3 Listening and connecting



```
sgt@sgt:~$ ls
6.828  Desktop    Downloads  Pictures  snap       Videos
code   Documents  Music      Public    Templates
sgt@sgt:~$ netcat -v -l -p 9090
Listening on 0.0.0.0 9090
Connection received on localhost 59982
11
23
4
hello
```

```
sgt@sgt:~/Desktop$ telnet localhost 9090
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
11
23
4
hello
^[^A
```

3. Writing a network program using an OS stream socket

3.1 Let's get started--fetching and building the starter code

1. 获取源码

```
sgt@sgt:~/cs144$ git clone https://github.com/cs144/minnow
Cloning into 'minnow'...
remote: Enumerating objects: 278, done.
remote: Counting objects: 100% (173/173), done.
remote: Compressing objects: 100% (91/91), done.
remote: Total 278 (delta 97), reused 82 (delta 82), pack-reused 105
Receiving objects: 100% (278/278), 110.96 KiB | 562.00 KiB/s, done.
Resolving deltas: 100% (135/135), done.
```

2. 建立个人仓库

3.2 Modern C++：mostly safe but still fast and low-level

3.3 Reading the Minnow support code

3.4 Writing webget

webget

```
void get_URL( const string& host, const string& path )
{
  TCPSocket socket;
  //建立连接
  socket.connect(Address(host,"http"));
  //发起请求（请求报文）
  socket.write("GET "+path+" HTTP/1.1\r\n");
  socket.write("HOST: "+host+"\r\n");
  socket.write("Connection: close\r\n");
  socket.write("\r\n");
  //写结束
  socket.shutdown(SHUT_WR);
  string buf;
  //读返回的字符
  while(!socket.eof()){
    socket.read(buf);
    cout<<buf;
  }
  //关闭连接
  socket.close();
}
```

编译过程

```
mkdir build
cd build
cmake ..
make
```

the output

```
HTTP/1.1 200 OK
Date: Thu, 07 Dec 2023 11:06:50 GMT
Server: Apache
Last-Modified: Thu, 13 Dec 2018 15:45:29 GMT
ETag: "e-57ce93446cb64"
Accept-Ranges: bytes
Content-Length: 14
Connection: close
Content-Type: text/plain

```

```
Hello, CS144!
```

输入make --build build --target check_webget

The result

```
Test project /home/sgt/cs144/minnow/build
    Start 1: compile with bug-checkers
1/2 Test #1: compile with bug-checkers ........   Passed    0.22 sec
    Start 2: t_webget
2/2 Test #2: t_webget .........................   Passed    1.09 sec

100% tests passed, 0 tests failed out of 2

Total Test time (real) =   1.32 sec
Built target check_webget
```

4. An in-memory reliable byte stream

一端读，一端写，使用队列。

byte_stream.hh

```cpp
class ByteStream
{
protected:
  uint64_t capacity_;
  std::deque<char> deque;
  uint64_t push_len=0;                          //已输入长度
  uint64_t pop_len=0;                           //以输出长度
  // Please add any additional state to the ByteStream here, and not to the Writer and Reader interfaces.
  bool write_state=false;  //the state of writer.
  bool error_state=false; //the state of state;
  ............
```

byte_stream.cc

```cpp
#include <stdexcept>

#include "byte_stream.hh"

using namespace std;

ByteStream::ByteStream( uint64_t capacity )
  : capacity_( capacity ), queue(), push_len( 0 ), pop_len( 0 ), write_state( false ), error_state( false )
{}
```

```cpp

void Writer::push( string data )
{
  for ( auto ch : data ) {
    if ( available_capacity() > 0 ) {
      queue.push( ch );
      push_len++;
    }
  }
  // Your code here.
  (void)data;
}

void Writer::close()
{
  // Your code here.
  write_state = true;
}

void Writer::set_error()
{
  // Your code here.
  error_state = true;
}

bool Writer::is_closed() const
{
  // Your code here.
  return write_state;
}

uint64_t Writer::available_capacity() const
{
  // Your code here.
  return capacity_ - queue.size();
}

uint64_t Writer::bytes_pushed() const
{
  // Your code here.
  return push_len;
}

string_view Reader::peek() const
{

  return string_view { &queue.front(), 1 };
}
```

```cpp

bool Reader::is_finished() const
{
  if ( bytes_buffered() == 0 && write_state ) {
    return true;
  }
  return false;
}

bool Reader::has_error() const
{
  // Your code here.

  return error_state;
}

void Reader::pop( uint64_t len )
{
  for ( uint64_t i = 0; i < len; i++ ) {
    queue.pop();
    pop_len++;
  }

  (void)len;
}

uint64_t Reader::bytes_buffered() const
{
  // Your code here.
  return queue.size();
}

uint64_t Reader::bytes_popped() const
{
  // Your code here.
  return pop_len;
}

```

输入 cmake --build build --target check0

the result

```
Test project /home/sgt/cs144/minnow/build
      Start  1: compile with bug-checkers
 1/10 Test  #1: compile with bug-checkers ........   Passed    7.41 sec
      Start  2: t_webget
 2/10 Test  #2: t_webget .........................   Passed    1.39 sec
```

```
      Start  3: byte_stream_basics
 3/10 Test  #3: byte_stream_basics ...............   Passed    0.01 sec
      Start  4: byte_stream_capacity
 4/10 Test  #4: byte_stream_capacity .............   Passed    0.01 sec
      Start  5: byte_stream_one_write
 5/10 Test  #5: byte_stream_one_write ............   Passed    0.02 sec
      Start  6: byte_stream_two_writes
 6/10 Test  #6: byte_stream_two_writes ...........   Passed    0.02 sec
      Start  7: byte_stream_many_writes
 7/10 Test  #7: byte_stream_many_writes ..........   Passed    0.07 sec
      Start  8: byte_stream_stress_test
 8/10 Test  #8: byte_stream_stress_test ..........   Passed    0.48 sec
      Start  9: compile with optimization
 9/10 Test  #9: compile with optimization ........   Passed    3.46 sec
      Start 10: byte_stream_speed_test
             ByteStream throughput: 0.46 Gbit/s
10/10 Test #10: byte_stream_speed_test ...........   Passed    0.36 sec

100% tests passed, 0 tests failed out of 10

Total Test time (real) =  13.23 sec
Built target check0
```