目录

1 实验目的与要求

1.1 实验目的

首先学习面向 TCP 连接的套接字编程基础知识：如何创建套接字，将其绑定到特定的地址和端口，以及发送和接收数据包。其次还将学习 HTTP 协议格式的相关知识。在此基础上，本实验开发一个简单的 Web 服务器，它仅能处理一个 HTTP 连接请求。

1.2 实验要求

Web 服务器的基本功能是接受并解析客户端的 HTTP 请求，然后从服务器的文件系统获取所请求的文件，生成一个由头部和响应文件内容所构成成的 HTTP 响应消息，并将该响应消息发送给客户端。如果请求的文件不存在于服务器中，则服务器应该向客户端发送“404 Not Found”差错报文。具体的过程和步骤分为：

1. 当一个客户（浏览器）连接时，创建一个连接套接字；
2. 从这个连接套接字接收 HTTP 请求；
3. 解释该请求以确定所请求的特定文件；
4. 从服务器的文件系统获得请求的文件；
5. 创建一个由请求的文件组成的 HTTP 响应报文，报文前面有首部行；
6. 经 TCP 连接向请求浏览器发送响应；
7. 如果浏览器请求一个在该服务器中不存在的文件，服务器应当返回一个“404 Not Found”差错报文。

2 实验原理与实验内容

2.1 实验原理

2.1.1 Socket 编程接口

要实现 Web 服务器，需使用套接字 Socket 编程接口来使用操作系统提供的网络通信功能。Socket 是应用层与 TCP/IP 协议族通信的中间软件抽象层，是一组编程接口。它把复杂的 TCP/IP 协议族隐藏在 Socket 接口后面，对用户来说，一组简单的接口就是全部，让 Socket 去组织数据，以符合指定的协议。使用 Socket 后，无需深入理解 TCP/UDP 协议细节（因为 Socket 已经为我们封装好了），只需要遵循 Socket 的规定去编程，写出的程序自然就是遵循 TCP/UDP 标准的。Socket 的地位如下图所示：

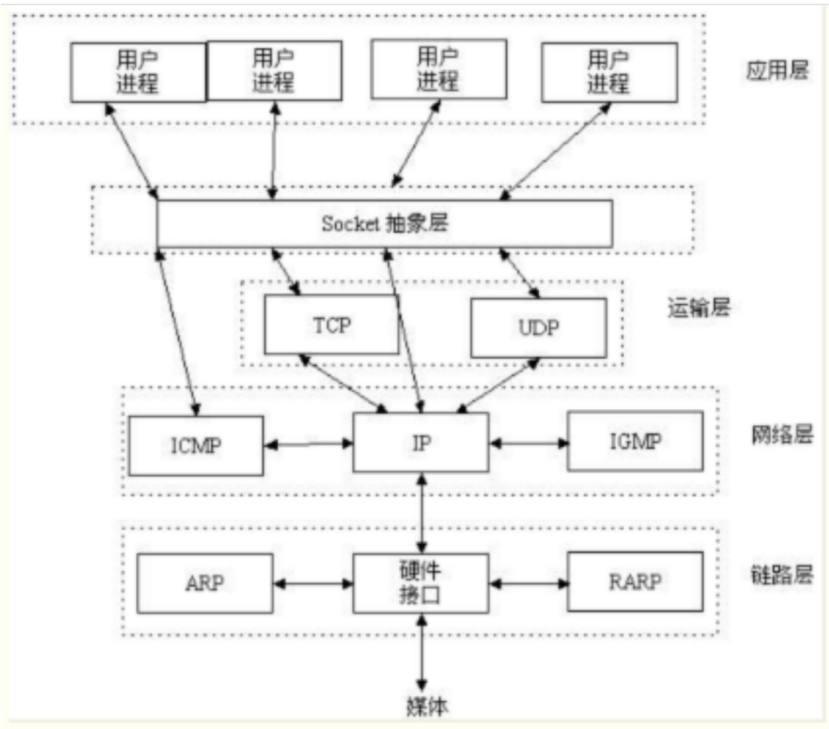

图 1: Socket 的地位

从某种意义上说，Socket 由地址 IP 和端口 Port 构成。IP 是用来标识互联网中的一台主机的位置，而 Port 是用来标识这台机器上的一个应用程序，IP 地址是配置到网卡上的，而 Port 是应用程序开启的，IP 与 Port 的绑定就标识了互联网中独一无二的一个应用程序。

套接字类型流式套接字 (SOCK_STREAM): 用于提供面向连接、可靠的数据传输服务。数据报套接字(SOCK_DGRAM): 提供了一种无连接的服务。该服务并不能保证数据传输的可靠性，数据有可能在传输过程中丢失或出现数据重复，且无法保证顺序地接收到数据。原始套接字 (SOCK_RAW): 主要用于实现自定义协议或底层网络协议。

在本 WEB 服务器程序实验中，采用流式套接字进行通信。其基本模型如下图所示:

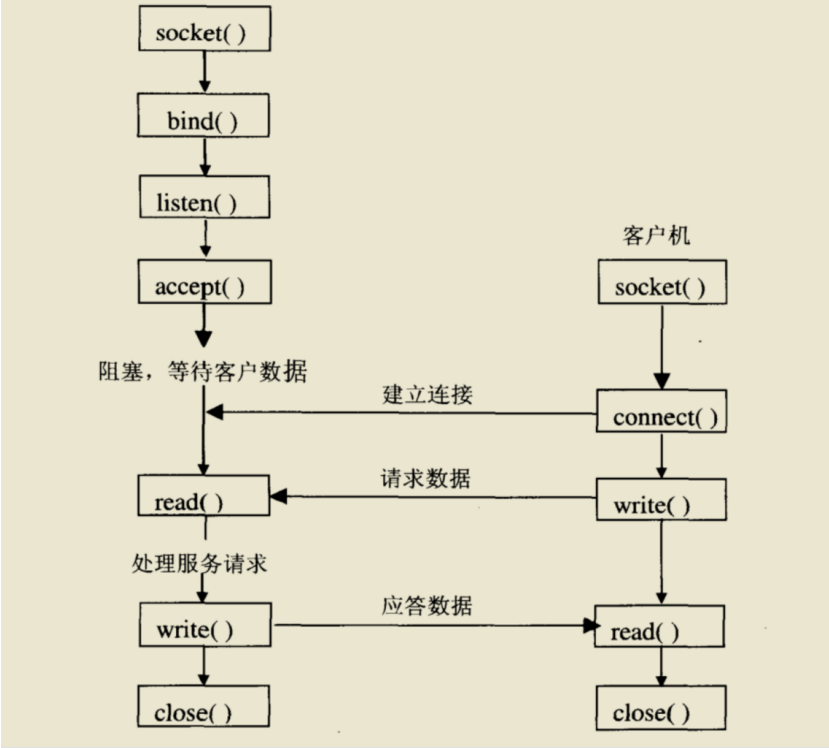

图 2: Socket 的基本模型

其工作过程如下：服务器首先启动，通过调用 socket () 建立一个套接字，然后调用绑定方法 bind () 将该套接字和本地网络地址联系在一起，再调用 listen () 使套接字做好侦听连接的准备，并设定的连接队列的长度。客户端在建立套接字后，就可调用连接方法 connect () 向服务器端提出连接请求。服务器端在监听到连接请求后，建立和该客户端的连接，并放入连接队列中，并通过调用 accept () 来返回该连接，以便后面通信使用。客户端和服务器连接一旦建立，就可以通过调用接收方法 recv () / recvfrom () 和发送方法 send () / sendto () 来发送和接收数据。最后，待数据传送结束后，双方调用 close () 关闭套接字。

2.1.2 HTTP 传输协议

超文本传输协议（HTTP）是用于 Web 上进行通信的协议：它定义 Web 浏览器如何从 Web 服务器请求资源以及服务器如何响应。为简单起见，在该实验中将处理 HTTP 协议的 1.0 版。HTTP 通信以事务形式进行，其中事务由客户端向服务器发送请求，然后读取响应组成。请求和响应消息共享一个通用的基本格式：

- 初始行（请求或响应行）
- 零个或多个头部行
- 空行（CRLF）
- 可选消息正文

对于大多数常见的 HTTP 事务，协议归结为一系列相对简单的步骤：

首先，客户端创建到服务器的连接；然后客户端通过向服务器发送一行文本来发出请求。这请求行包 HTTP 方法（比如 GET，POST、PUT 等），请求 URI（类似于 URL），以及客户机希望使用的协议版本（比如 HTTP/1.0）；接着，服务器发送响应消息，其初始行由状态线（指示请求是否成功），响应状态码（指示请求是否成功完成的数值），以及推理短语（一

种提供状态代码描述的英文消息组成)；最后一旦服务器将响应返回给客户端，它就会关闭连接。

2.2 实验内容

建立一个简单的 Web 服务器端，它能够接收客户端的请求，解析请求的方法和路径，然后返回相应的文件内容作为响应。

3 实验具体设计实现及结果

3.1 实验具体设计实现

3.1.1 程序结构图

根据实验相关要求，设计了本程序，程序的框架图如下所示：

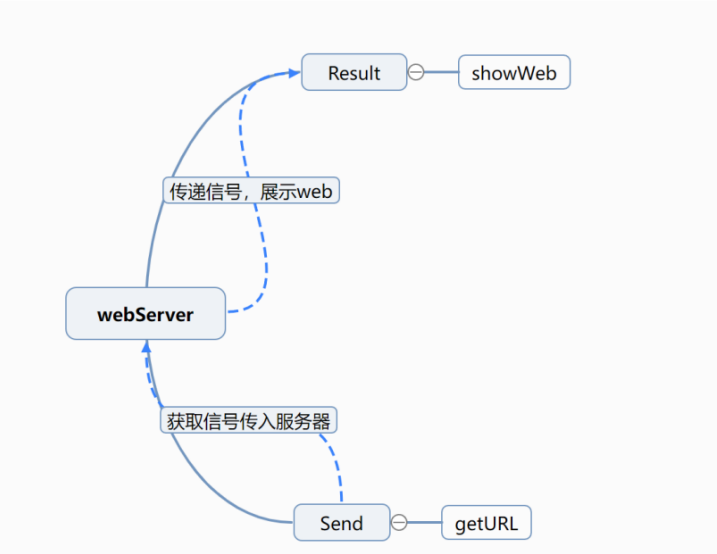

图 3: 程序结构图

在 webServer 中，使用 Socket 循环以等待客户端的请求。本程序采用 Send 类中的 getURL 函数，将读取的数据转换为字符串，然后进行两次分

割，以获取用户在浏览器上发出的相关请求。在获得请求后，通过 Result 类，实现 http 报文的发送，以及 web 页面的展示。

3.1.2 webServer 类

该类中，首先自定义使用的端口号为 1206，然后创建 ServerSocket 对象，用以创建服务器套接字并指定端口号。

接下来循环等待客户端的请求，若客户端发来请求，则程序监听并接收客户端的连接请求，并获取与客户端通信的输入输出刘。

然后创建 Send 对象，用以处理客户端请求，获取 HTML 页面名称；创建 Result 对象，用以处理并返回 HTML 页面，展示 HTML 页面内容，并发送给客户端。

实现代码见附录.

3.1.3 Send 类

首先，从客户端接收请求报文，接收最多 1024 个字节的数据，解码成字符串，存储在 byte 数组中；

然后，将请求报文按照回车换行符分割成请求行和请求头部，请求行中包含请求方法、请求资源路径和 HTTP 版本号，请求头部中包含请求的其他信息；然后再进行第二次分割，分割后的结果即客户端请求的 URL

以下是实现代码：

```
1   import java.io.IOException;
import java.io.InputStream;
3
public class  Send {
5     InputStream input;
      //int port;
7
      public Send(InputStream input) {
9         //this.port = port;
          this.input = input;
```

```
    }

    // 获取客户端请求的URL
    public String getURL(){
        byte[] by = new byte[1024];
        try {
            input.read(by); // 从输入流中读取客户端请求的数据
        } catch (IOException e) {
            e.printStackTrace();
        }
        String s  = new String(by); // 将读取的数据转换为字符串
        //System.out.println(s);
        String ss1[] = s.split("/"); // 以"/"字符为分隔符将字符串分割成多个部分
        if(ss1.length >= 2){
            if((ss1[1] != null)){
                String ss2[] = ss1[1].split(" "); // 将第一个分割部分以空格为分隔符再次分割
                return ss2[0]; // 返回第二次分割后的结果，即客户端请求的URL
            }
        }
        return "-1"; // 如果无法解析出URL，则返回"-1"
    }
}
```

Send 类

3.1.4 Result 类

Result 类的函数流程图如下所示：

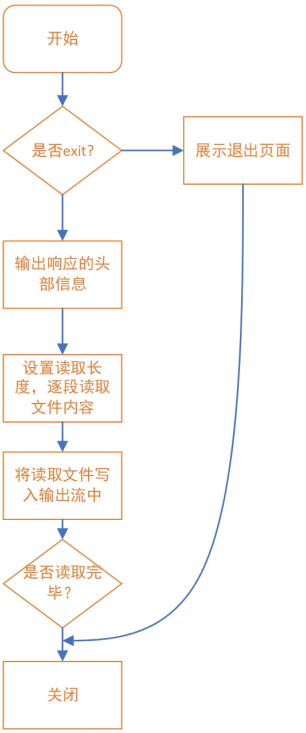

图 4: Result 流程图

Result 类用以判断请求并返回结果。如果请求为 exit，则退出相关程序，若为其余请求，则根据请求类型进行处理

当请求不为 exit 时，根据文件名返回对应的文件内容。当文件存在且为普通文件时，先获取文件的扩展名，再输出响应的头部信息，包括状态码、Content-Type 和 Content-Length，然后逐个读取文件内容，并写入输出流中。

以下是实现代码:

```
import java.io.*;

public class Result {
    private OutputStream os;
    private String fileName;

```

```
    public Result(OutputStream os, String fileName) {
        this.os = os;
        this.fileName = fileName;
    }

    public void showWeb() throws IOException {
        if (fileName.equals("exit")) {
            // 如果文件名为"exit", 返回一个退出页面
            PrintStream pt = new PrintStream(os);
            pt.println("HTTP/1.1 200 OK");
            pt.println("Content-Type: text/html;charset=UTF-8");
            pt.println("Content-Length: 32");
            pt.println("");
            pt.println("<h1>页面已经退出了</h1>");
        } else {
            // 否则根据文件名返回相应的文件内容
            File file = new File("C:\\ComputerNetwork\\A3\\src\\
                " + fileName);
            if (file.exists() && file.isFile()) {
                // 如果文件存在且为普通文件
                String extension = getFileExtension(fileName);
                String contentType = getContentType(extension);

                PrintStream ps = new PrintStream(os);
                //输出响应的头部信息, 包括状态码、Content-Type和
                    Content-Length。
                ps.println("HTTP/1.1 200 OK");
                ps.println("Content-Type: " + contentType);
                ps.println("Content-Length: " + file.length());
                ps.println("");
                FileInputStream fis = new FileInputStream(file);
                byte[] buffer = new byte[4096];
                int bytesRead;
                while ((bytesRead = fis.read(buffer)) != -1) {
                    // 逐个读取文件内容并写入输出流中
```

```
40                      os.write(buffer, 0, bytesRead);
                    }
42                  fis.close();
                } else {
44                  // 如果文件不存在, 返回404错误页面
                    PrintStream pt = new PrintStream(os);
46                  pt.println("HTTP/1.1 404 File Not Found");
                    pt.println("Content-Type: text/html");
48                  pt.println("Content-Length: 23");
                    pt.println("");
50                  pt.println("<h1>File Not Found</h1>");
                }
52          }
        }
54
        private String getFileExtension(String fileName) {
56          // 获取文件的扩展名
            int dotIndex = fileName.lastIndexOf(".");//获取最后一个
                "."字符的索引位置
58          if (dotIndex != -1 && dotIndex < fileName.length() - 1)
                {
                return fileName.substring(dotIndex + 1);
60          }
            return "";
62      }

64      private String getContentType(String fileExtension) {
            // 根据扩展名返回相应的Content-Type
66          switch (fileExtension) {
                case "html":
68                  return "text/html";
                case "png":
70                  return "image/png";
                case "jpg":
72                  return "image/jpeg";
```

```
            default:
                return "application/octet-stream";
        }
    }
}
```

Result 类

3.2 实验结果

3.2.1 启动后控制台输出

图 5: 启动后控制台输出

3.2.2 浏览器访问结果

图 6: 浏览器访问结果

3.2.3 浏览器访问 test.html 结果

图 7: 浏览器访问 test.html 结果

3.2.4 浏览器访问不存在的文件结果

图 8: 浏览器访问不存在的文件结果

3.2.5 使用错误方法访问结果

图 9: 使用错误方法访问结果

4 实验设备与实验环境

1. 编程语言：Java

2. 编程环境：IDEA，windows10 操作系统

5 实验总结

在本次实验中，程序实现了一个简单的 Web 服务器端。该服务器能够接收客户端的请求，解析请求的方法和路径，然后返回相应的文件内容作为响应。实现过程中遇到的困难：在这个实验中，主要的困难是理解和处理 HTTP 协议相关的内容。需要熟悉 HTTP 请求和响应的格式以及常见的状态码。另外，还需要处理文件的读取和异常情况。

通过这个实验，我对基本的 Web 服务器端实现有了更深入的理解。我学会了使用 Python 的 socket 库创建服务器套接字、绑定地址和端口，监听客户端连接请求，并能够处理 HTTP 请求和构造响应报文。

这个简单的 Web 服务器端实现还有很大的改进空间。例如，可以添加更多的 HTTP 方法的支持，如 POST、PUT 等。还可以引入并发处理多个客户端连接的能力，提高服务器的并发性能。此外，还可以考虑对请求参数进行解析和处理，增加服务器的功能和灵活性。

总体而言，这个实验帮助我更好地理解了 Web 服务器端的基本原理和实现方式。通过实际编码和调试的过程，我对 HTTP 协议和 socket 编程有了更深入的了解，为以后进一步探索网络编程和 Web 开发打下了坚实的基础。

6 附录

6.1 webServer 源码

```
    # 实验A.3： 简单Web服务器端实现

import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.net.ServerSocket;
import java.net.Socket;

public class webServer {
    public static void main(String[] args) {
        int port = 1206;
        ServerSocket serverSocket = null;
        try {
            // 创建服务器套接字并指定端口号
            serverSocket = new ServerSocket(port);
        } catch (Exception e) {
            e.printStackTrace();
            System.out.println("输入输出有误！！！");
        }
        while (true) { // 循环等待客户端请求
            Socket socket = null;
            InputStream is = null;
            OutputStream os = null;
            try {
                // 监听并接受客户端连接请求
                socket = serverSocket.accept();
                // 获取与客户端通信的输入输出流
                is = socket.getInputStream();
                os = socket.getOutputStream();
            } catch (Exception e) {
                e.printStackTrace();
```

```
32                  System.out.println("消息传输有误！！");
                }
34              // 创建一个Send对象，用于处理客户端请求
                Send send = new Send(is);
36              // 获取请求的HTML页面名称
                String htmlName = send.getURL();
38              // 创建一个结果对象，用于处理并返回HTML页面
                Result result = new Result(os, htmlName);
40              try {
                    // 显示HTML页面内容并发送给客户端
42                  result.showWeb();
                } catch (IOException e) {
44                  e.printStackTrace();
                }
46              if (htmlName.equals("exit") || htmlName.equals("-1")
                    ) {
                    System.out.println("结束");
48                  break;
                }
50          }
        }
52  }

54  import java.io.IOException;
    import java.io.InputStream;
56
    public class  Send {
58      InputStream input;
        //int port;
60
        public Send(InputStream input) {
62          //this.port = port;
            this.input = input;
64      }
```

```
66      // 获取客户端请求的URL
        public String getURL(){
68          byte[] by = new byte[1024];
            try {
70              input.read(by); // 从输入流中读取客户端请求的数据
            } catch (IOException e) {
72              e.printStackTrace();
            }
74          String s  = new String(by); // 将读取的数据转换为字符串
            //System.out.println(s);
76          String ss1[] = s.split("/"); // 以"/"字符为分隔符将字符
                串分割成多个部分
            if(ss1.length >= 2){
78              if((ss1[1] != null)){
                    String ss2[] = ss1[1].split(" "); // 将第一个分
                        割部分以空格为分隔符再次分割
80                  return ss2[0]; // 返回第二次分割后的结果, 即客户
                        端请求的URL
                }
82          }
            return "-1"; // 如果无法解析出URL, 则返回"-1"
84      }
    }
86
    import java.io.*;
88
    public class Result {
90      private OutputStream os;
        private String fileName;
92
        public Result(OutputStream os, String fileName) {
94          this.os = os;
            this.fileName = fileName;
96      }
```

```
98      public void showWeb() throws IOException {
            if (fileName.equals("exit")) {
100             // 如果文件名为"exit", 返回一个退出页面
                PrintStream pt = new PrintStream(os);
102             pt.println("HTTP/1.1 200 OK");
                pt.println("Content-Type: text/html;charset=UTF-8");
104             pt.println("Content-Length: 32");
                pt.println("");
106             pt.println("<h1>页面已经退出了</h1>");
            } else {
108             // 否则根据文件名返回相应的文件内容
                File file = new File("C:\\ComputerNetwork\\A3\\src\\
                    " + fileName);
110             if (file.exists() && file.isFile()) {
                    // 如果文件存在且为普通文件
112                 String extension = getFileExtension(fileName);
                    String contentType = getContentType(extension);
114
                    PrintStream ps = new PrintStream(os);
116                 //输出响应的头部信息, 包括状态码、Content-Type和
                        Content-Length。
                    ps.println("HTTP/1.1 200 OK");
118                 ps.println("Content-Type: " + contentType);
                    ps.println("Content-Length: " + file.length());
120                 ps.println("");
                    FileInputStream fis = new FileInputStream(file);
122                 byte[] buffer = new byte[4096];
                    int bytesRead;
124                 while ((bytesRead = fis.read(buffer)) != -1) {
                        // 逐个读取文件内容并写入输出流中
126                     os.write(buffer, 0, bytesRead);
                    }
128                 fis.close();
                } else {
130                 // 如果文件不存在, 返回404错误页面
```

```
                PrintStream pt = new PrintStream(os);
132             pt.println("HTTP/1.1 404 File Not Found");
                pt.println("Content-Type: text/html");
134             pt.println("Content-Length: 23");
                pt.println("");
136             pt.println("<h1>File Not Found</h1>");
            }
138     }
    }
140
    private String getFileExtension(String fileName) {
142     // 获取文件的扩展名
        int dotIndex = fileName.lastIndexOf(".");//获取最后一个
            "."字符的索引位置
144     if (dotIndex != -1 && dotIndex < fileName.length() - 1)
            {
            return fileName.substring(dotIndex + 1);
146     }
        return "";
148 }

150 private String getContentType(String fileExtension) {
        // 根据扩展名返回相应的Content-Type
152     switch (fileExtension) {
            case "html":
154             return "text/html";
            case "png":
156             return "image/png";
            case "jpg":
158             return "image/jpeg";
            default:
160             return "application/octet-stream";
        }
162 }
}
```

6.2 nav.html 源码

```
<!DOCTYPE html>
<html>
<head>
  <style>
      body {
          margin: 0;
          padding: 0;
          background-size: cover;
          background-repeat: no-repeat;
          background-image: url('background.jpg');
      }

      .container {
          position: absolute;
          top: 50%;
          left: 50%;
          transform: translate(-50%, -50%);
          text-align: center;
      }

      .text {
          font-size: 24px;
          color: black;
          text-decoration: none;
          transition: color 0.5s;
      }

      .text:hover {
          color: blue;
```

```
        }
    </style>
</head>
<body>
<div class="container">
    <img src="background.png" alt="Background Image">
    <a href="#" class="text">The A3 experiment of Zwr with
        8207211004</a>
    <p id="time" class="text"></p>
</div>

<script>
    function updateTime() {
        var timeElement = document.getElementById("time");
        var currentTime = new Date();
        var year = currentTime.getFullYear();
        var month = currentTime.getMonth() + 1;
        var day = currentTime.getDate();
        var hours = currentTime.getHours();
        var minutes = currentTime.getMinutes();
        var seconds = currentTime.getSeconds();

        // 将单个数字的月份、日期、小时、分钟和秒数转换为两位数
        month = (month < 10 ? "0" : "") + month;
        day = (day < 10 ? "0" : "") + day;
        hours = (hours < 10 ? "0" : "") + hours;
        minutes = (minutes < 10 ? "0" : "") + minutes;
        seconds = (seconds < 10 ? "0" : "") + seconds;

        var timeString = year + "-" + month + "-" + day + " " +
            hours + ":" + minutes + ":" + seconds;
        timeElement.textContent = timeString;

        // 每秒钟更新一次时间
        setTimeout(updateTime, 1000);
```

```
63      }
65      updateTime();
    </script>
67  </body>
    </html>
```

nav.html

6.3 test.html 源码

```
1       <!DOCTYPE html>
        <html lang="en">
3       <head>
            <meta charset="UTF-8">
5           <title>About Us</title>
            <style>
7               body{
                    text-align: center;
9               }
                a{
11                  text-decoration: none;
                }
13              a:hover{
                    background: #0000ff;
15                  color: #fff;
                }
17          </style>
        </head>
19      <body>
            <h1>This is JiangWu,aka J&Ocean</h1>
21          <h2>a handsome boy from CSU,major in COMPUTER SCIENCE
                AND TECHNOLOGY</h2>
            <a href="index.html">Back To The Homepage</a>
23          <br>
```

```
        <br>
25      <a href="Contactus.html">Contact Us</a>
        <br>
27      <p>This is the AboutUs page</p>
    </body>
29  </html>
```

test.html