Socket网络编程

目录

Socket简介

什么是socket

电器 → socket → 电网

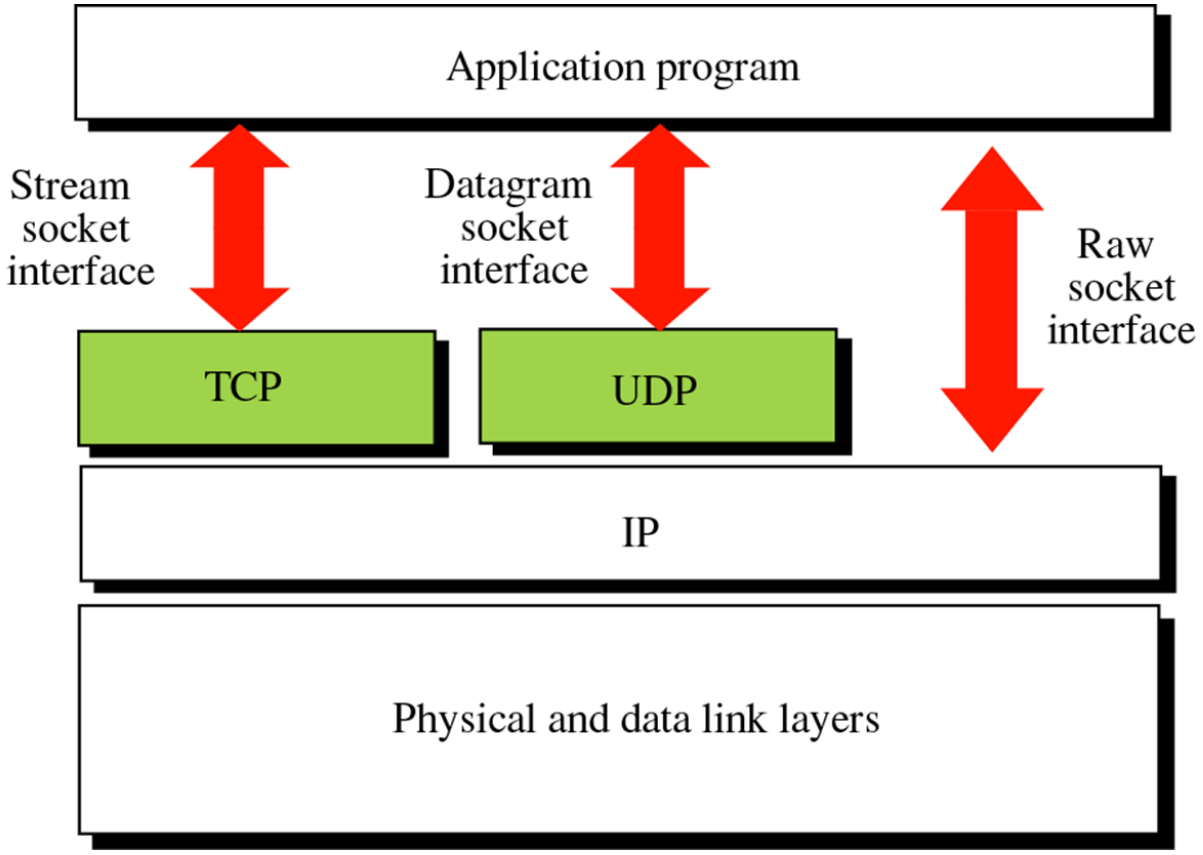
应用程序 → socket → 计算机网络

什么是socket

Socket是供应用程序访问传输层功能的一组API
由应用程序创建
- 提供了两种类型的通信接口
  - 可靠的,面向连接的Socket
  - 不可靠的,面向消息的Socket

通过socket接口应用程序可以:
- 向网络上的应用**发送数据**
- **接收**其它应用发来的**数据**

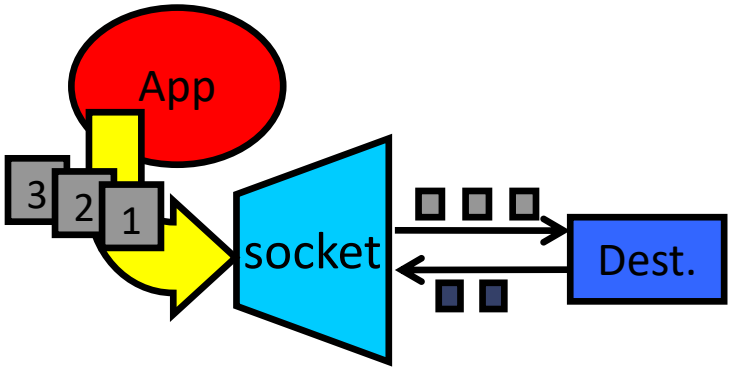
Socket接口在网络协议栈中的位置

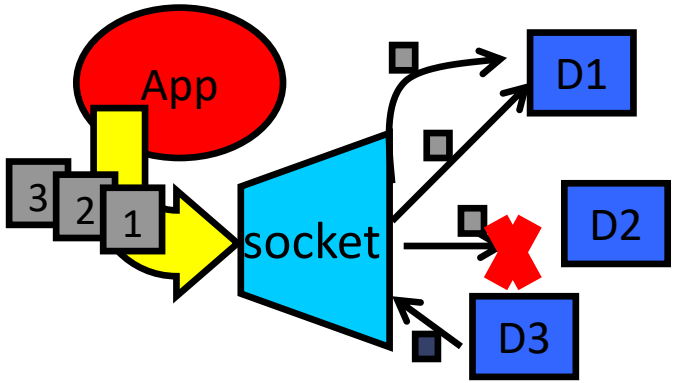


两种类型的socket

SOCK_STREAM
- 基于TCP协议
- 可靠数据传输
- 面向连接
- 全双工

SOCK_DGRAM
- 基于UDP协议
- 不可靠数据传输
- 面向消息(没有连接)
- 能够发送或接收数据



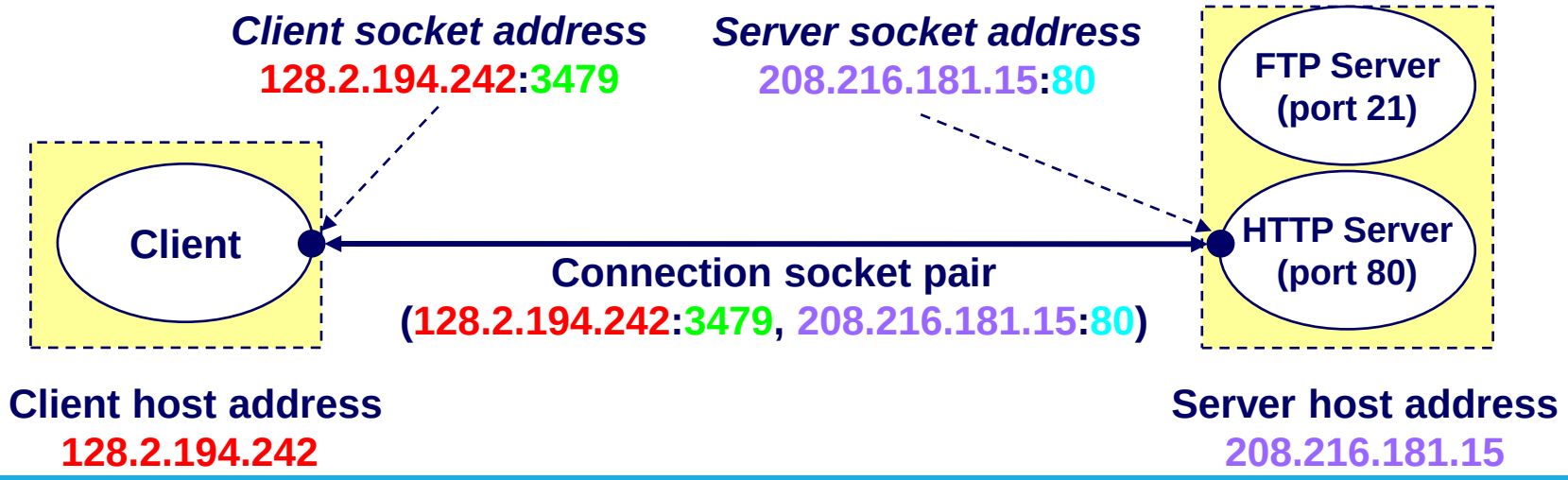


Socket如何寻找通讯对象

IP地址（IP Address）
- 数据发送到哪个主机。 (Addressing)

端口号（Port）
- 数据发送到主机中的哪个进程。 (Multiplexing)

Socket = IP地址+端口号

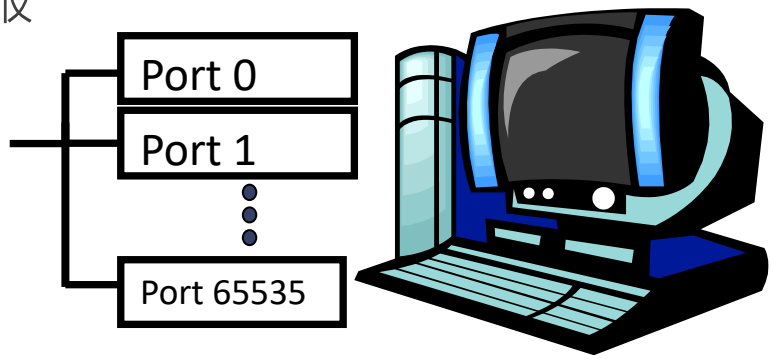


关于端口号

每个主机有65536个端口号（2^16）

其中一些端口号被保留给常用的程序和服务，被称为"熟知端口号

- 20 21: FTP
- 23: Telnet
- 80: HTTP
- 443：HTTPs



http://www.tju.edu.cn/

http://www.tju.edu.cn:80/

https://www.baidu.com:443

https://www.baidu.com

在写其它的服务程序时应该避开这些端口，通常从1024开始

Socket API

不同操作系统提供的 socket API.

- Berkeley UNIX: BSD Socket
- Microsoft Windows : WINSOCK (Windows Socket Interface)
- AT&T UNIX System V: TLI (Transport Layer Interface)

不同编程语言提供的socket API

- 高级语言的socket API通常是对操作系统socket API的二次封装
- 虽然不同操作系统、不同编程语言提供的socket API各有不同，但是其功能和使用方法都是类似的。

使用Socket API进行网络编程

如何使用socket API

初 始 化

socket() 方法
设置远程主机在网络中的哪个位置 (IP address, hostname)
设置应该把数据传送到远程主机的哪个程序/服务 (port)

收发数据

和其他的I/O操作类似
send <--> write
recv <--> read

关 闭

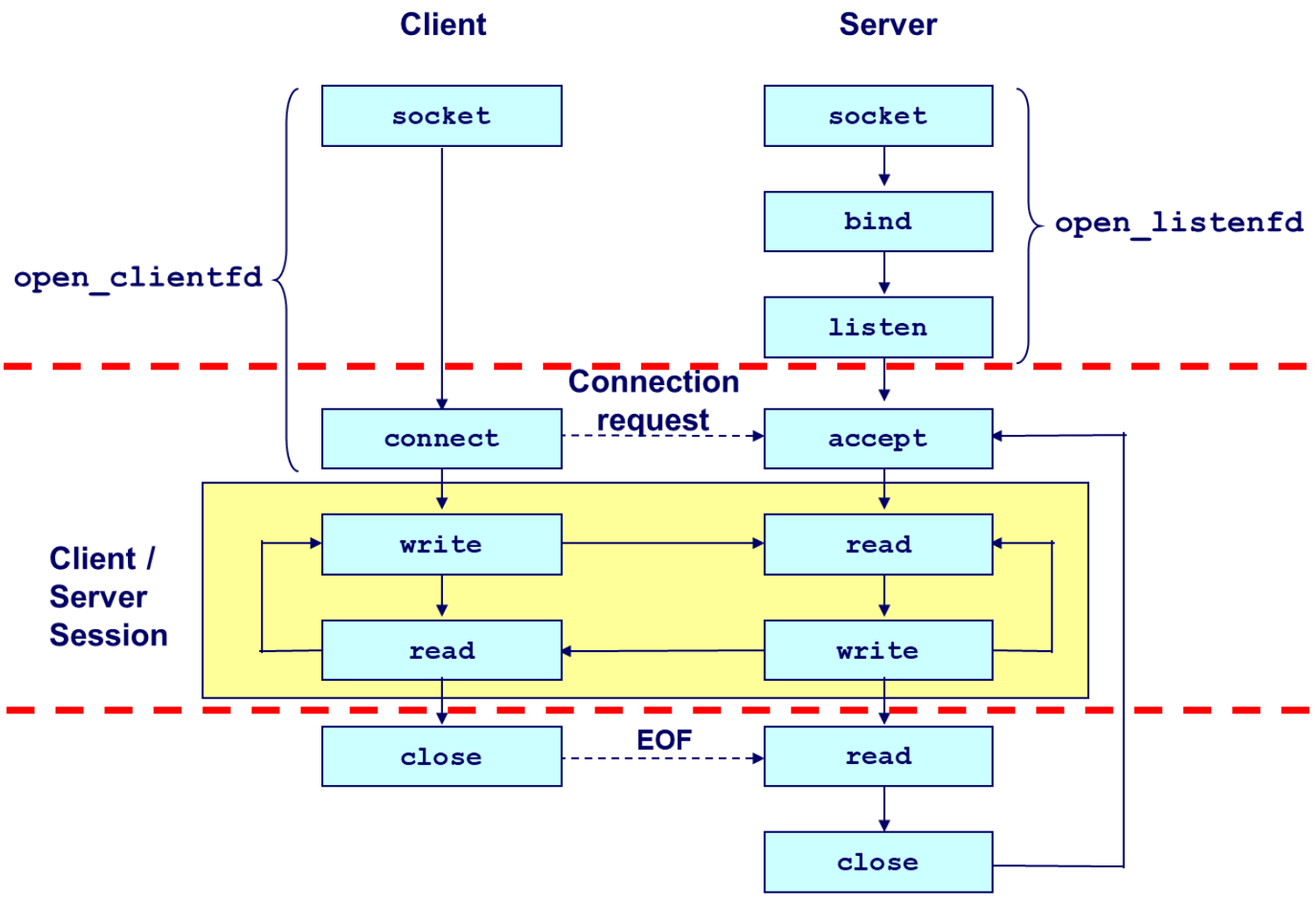
close() 方法
退出server程序
解除端口占用

TCP socket

初始化

收发数据

关闭



初始化(创建、绑定与监听)

- 创建

sock = socket(addr_family, type)

addr_family: ip地址类型

AF_INET	IPv4
AF_INET6	IPv6

type: socket类型

SOCK_STREAM	TCP
SOCK_DGRAM	UDP

- 绑定

bind(sock, addr, addrlen)

sock: socket函数调用返回的套接字

addr: 地址结构体，调用bind之后这个地址与参数sockfd指定的套接字关联

addrlen: addr的长度

初始化(创建、绑定与监听)

- 监听

listen(sock, backlog)

scok: socket函数调用返回的套接字

backlog: 指定排列在队列中的最大的连接数量,不得小于1.

至此,服务端已经完成初始化操作

客户端的初始化只需要调用socket()方法

不需要进行绑定和监听

收发数据(建立连接)

- 客户端:发送连接请求

```
connect(sock, server_addr, server_ addrlen)
```

scok: socket函数调用返回的套接字
server_addr: 服务端的地址
server_ addrlen: 服务端地址的长度

注意:
服务端被动等待连接
客户端主动发起连接

- 服务端:接受连接请求

accept方法是阻塞的
并且可以重复调用

```
conn = accept(sock, client_addr, clinet_ addrlen)
```

conn: 新的socket对象，用来在新建的连接上发送和接收数据
client_addr: 客户端的地址
client_ addrlen: 客户端地址的长度

收发数据

- 发送数据

```
count = send(sock, data, data_len, flag)
```

sock:发送数据的套接字
data: 要发送的数据
data_len: 数据的字节数
flag: 一组指定调用方式的标志，一般设置为0

- 接收数据

```
count = recv(sock, buf, buf_size, flag)
```

sock: 接收数据的套接字
buf: 接收数据的缓存
buf_size: 缓存大小
flag: 一组指定调用方式的标志，一般设置为0

注意：
recv方法默认是阻塞的
但是可以配置非阻塞

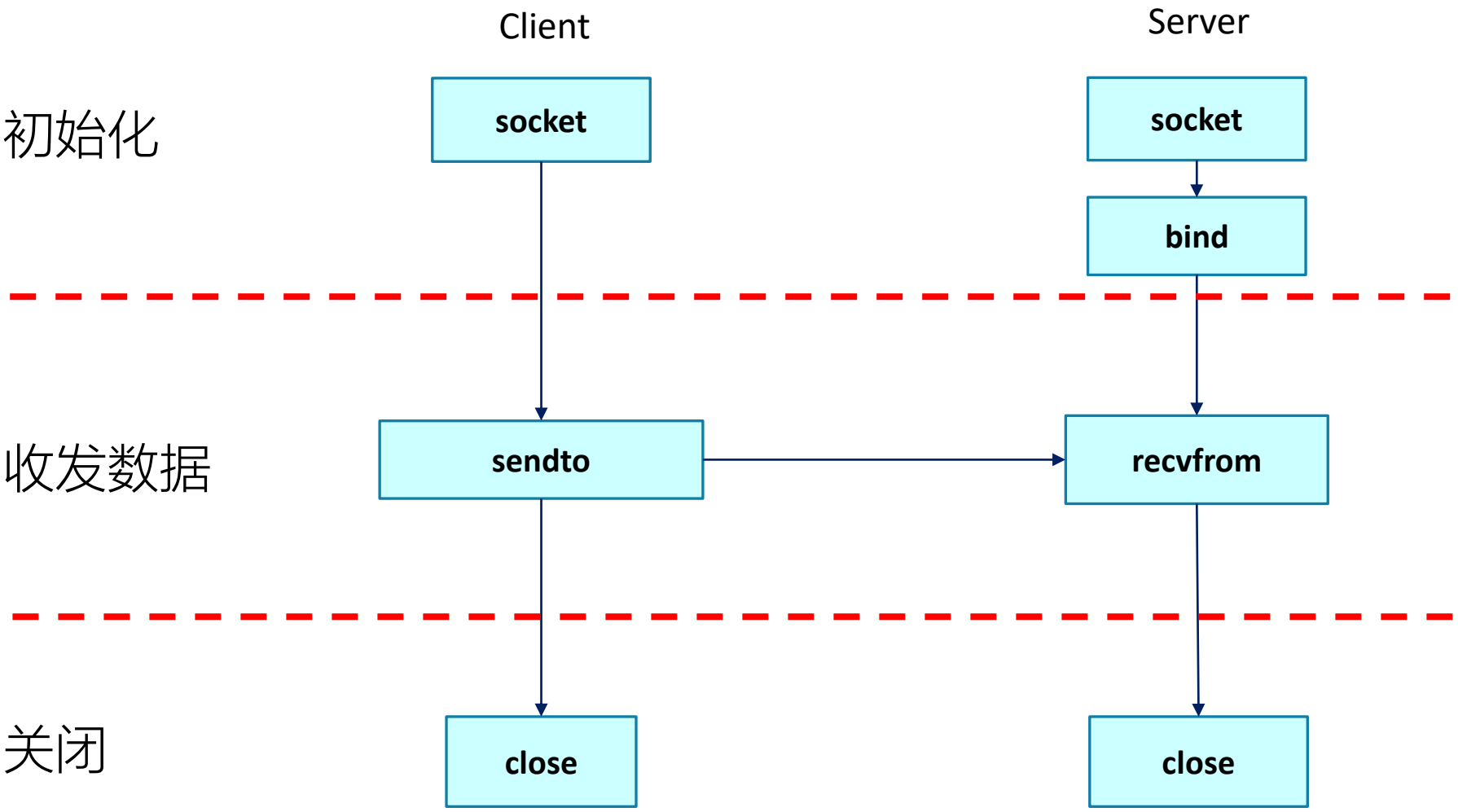
关闭

close(sock)

关闭socket

关闭之后该socket对象的所有操作将不可用

UDP socket



不需要listen connect accept

收发数据

- 发送数据

sendto(sock, buf, len, flag, dest_addr, addrlen)

sock:是由socket()调用返回的并且未作连接的套接字描述符（套接字号）

buf：发送缓冲区，往往是使用者定义的数组，该数组装有要发送的数据

len：发送缓冲区的大小，单位是字节

flag：一组指定调用方式的标志，一般设置为0

dest_addr:指向接收数据的主机地址信息的结构体，也就是该参数指定数据要发送到哪个主机哪个进程

UDP Socket

收发数据

- 接收数据

recvfrom(sock, buf, buf_size, flag, src_addr, addrlen)

sock:是由socket()调用返回的并且未作连接的套接字描述符（套接字号）

buf：接收缓冲区，往往是使用者定义的数组，该数组装有接收到的数据

len：接收缓冲区的大小，单位是字节

flag：一组指定调用方式的标志，一般设置为0

src_addr:指向发送数据的主机地址信息的结构体，也就是我们可以从该参数获取到数据是谁发出的

注意：
recv方法默认是阻塞的
但是可以配置非阻塞

Socket API总结

Sockets
- socket setup
- I/O
  - send & recv
  - sendto & recvfrom
- Close

TCP
- Client: socket()->connect()->send()/recv()->close()
- Server: socket()->bind()->listen()->accept()->send()/recv()->close()

UDP
- Client: socket()->sendto->close()
- Server: socket()->bind()->recvfrom()->close()

Socket API 网络编程基础代码

基础代码目录说明

.../echo_client.c - Simple echo network client
.../echo_server.c - Simple echo network server
.../Makefile - Contains rules for make

.../example.c - Example driver for parsing
.../lexer.l - Lex/Yacc related logic
.../parse.y
.../parse.c
.../parse.h

.../sample_request_simple - Example HTTP requests
.../sample_request_realistic

Socket编程实验作业

目标

- 实现HTTP/1.1的数据包解析
- 实现HTTP/1.1的HEAD、GET、POST方法
- 实现HTTP的并行请求和响应
- 实现多个客户端的并发连接

要求

- Linux平台、C语言
- 2人一组，每组独立完成
- 禁止抄袭

具体实验内容见

- Socket编程实践指导书

Socket编程实验作业

实验安排:

- **项目开始**

  项目于x月x日正式开始

- **工作进度报告**

  项目开始后，须在规定时间

  在智慧树平台提交工作进度报告

Socket编程实验作业

关键时间节点：

x月x日：项目启动

x月x日： 提交第1周进度报告。

x月x日：提交第2周进度报告。

x月x日：提交第3周进度报告。

x月x日：提交第4周进度报告。

x月x日：提交实验报告和最终源码。