## 视频内容

1. ICP\IP基础
2. socket编程
3. 进程间通信
4. 实战

## TCP\IP基础

1. OSI七层参考模型（理论标准） & TCP\IP四层模型（实际标准）

|  |  |
| --- | --- |
| 应用层 | http、ftp、pop3 |
| 表示层 | 数据压缩、加密等 |
| 会话层 | 管理会话 |
| 传输层 | tcp\udp，段 |
| 网络层 | Ip,icmp,igmp,报 |
| 数据链路层 | Frame 帧 |
| 物理层 | Byte |

OSI七层模型

|  |  |
| --- | --- |
| 应用层 |  |
| 传输层 |  |
| 网络层 |  |
| 链路层 | Arp、rarp |

TCP/IP四层模型

1. 概念：对等通信、封装、解封、端口

MTU:最大传输单元

路径MTU，传输路径上最小MTU

1. ping过程
   1. 根据域名到dns查询ip
   2. 在本地缓存查询ip对应的mac地址，若存在，直接在icmp中填充目的mac地址，然后发送icmp报文；否则，使用arp广播
   3. 符合目的ip的主机先将源目的ip和源mac地址存到自身缓存，填充arp报文的目的mac地址，返回给发送方
   4. 发送方收到回复，缓存目的ip的对应的mac，使用ping的icmp报文发送
2. IP头
3. TCP头
4. UDP头

## socket编程

1. 地址结构
   1. IPv4地址结构

struct sockaddr\_in **{**

uint8\_t sin\_len**;**

sa\_fimaly\_t sin\_family**;**

in\_port\_t sin\_port**;**

struct in\_addr sin\_addr**;**

char sin\_zone**[**8**];**

**}**

* 1. 通用地址结构

struct sockaddr**{**

uint8\_t sin\_len**;**

sa\_fimaly\_t sin\_family**;**

char sa\_data**[**14**];**

**}**

1. 大小端转换

头文件<arpa/inet.h>

unsigned int i **=** 0x12345678**;**

char**\*** p **=** **reinterpret\_cast<**char**\*>(&**i**);**

printf**(**"%0x %0x %0x %0x\n"**,**p**[**0**],**p**[**1**],**p**[**2**],**p**[**3**]);**//输出78 56 34 12

i **=** htonl**(**i**);**

p **=** **reinterpret\_cast<**char**\*>(&**i**);**

printf**(**"%0x %0x %0x %0x\n"**,**p**[**0**],**p**[**1**],**p**[**2**],**p**[**3**]);**//输出12 34 56 78

1. 点分法地址和网络地址转换函数

头文件包含<netinet/in.h>

**typedef** uint32\_t in\_addr\_t**;**

struct in\_addr **{**

in\_addr\_t s\_addr**;**

**};**

int inet\_aton**(**const char**\*** p**,** struct in\_addr **\*** inp**);**//0表示出错，1表示成功

char**\*** inet\_ntoa**(**truct in\_addr in**);**//返回点分ip

in\_addr\_t inet\_addr**(**const char **\***p**);**//返回uint32

char**\*** p **=** "192.168.188.146"**;**

struct in\_addr in**;**

**if(**inet\_aton**(**p**,&**in**)** **==** 0 **){**

printf**(**"%s\n"**,**"error"**);**

**}**

printf**(**"%u\n%u\n%s\n"**,**in**.**s\_addr**,**inet\_addr**(**p**),**inet\_ntoa**(**in**));**

1. socket回显程序
   1. 客户端：

#include<unistd.h>

#include<sys/socket.h>

#include<sys/types.h>

#include<arpa/inet.h>

#include<netinet/in.h>

#include<stdio.h>

#include<string.h>

#include<stdlib.h>

#include<assert.h>

int main**(){**

int sock **=** 0**;**

assert**((**sock **=** socket**(**AF\_INET**,**SOCK\_STREAM**,**0**))** **!=** **-**1**);**

int n**;**

setsockopt**(**sock**,**SOL\_SOCKET**,**SO\_REUSEADDR**,&**n**,sizeof(**n**));**

struct sockaddr\_in addr\_srv**;**

addr\_srv**.**sin\_family **=** AF\_INET**;**

addr\_srv**.**sin\_port **=** htons**(**8000**);**

addr\_srv**.**sin\_addr**.**s\_addr **=** inet\_addr**(**"127.0.0.1"**);**

assert**(**connect**(**sock**,(**struct sockaddr**\*)&**addr\_srv**,sizeof(**addr\_srv**))** **!=** **-**1**);**

char revbuf**[**1024**]** **=** **{**0**};**

char sedbuf**[**1024**]** **=** **{**0**};**

int ret **=** 0**;**

**while((**fgets**(**sedbuf**,sizeof(**sedbuf**),**stdin**))** **!=** **NULL){**

write**(**sock**,**sedbuf**,**strlen**(**sedbuf**));**

assert**((**ret **=** read**(**sock**,&**revbuf**,sizeof(**revbuf**)))** **!=** **-**1**);**

**if(**ret **==** 0**){**

printf**(**"peer closed\n"**);**

**break;**

**}**

printf**(**"%s"**,**revbuf**);**

memset**(&**revbuf**,**0**,sizeof(**revbuf**));**

memset**(&**sedbuf**,**0**,sizeof(**sedbuf**));**

**}**

close**(**sock**);**

**return** 0**;**

**}**

* 1. 服务端：（可接受多客户端连接）

#include<unistd.h>

#include<sys/socket.h>

#include<sys/types.h>

#include<arpa/inet.h>

#include<netinet/in.h>

#include<stdio.h>

#include<string.h>

#include<stdlib.h>

#include<assert.h>

void do\_server**(**int conn**){**

char revbuf**[**1024**]** **=** **{**0**};**

int ret **=** 0**;**

**while(**1**){**

memset**(&**revbuf**,**0**,sizeof(**revbuf**));**

assert**((**ret **=** read**(**conn**,**revbuf**,sizeof(**revbuf**)))** **!=** **-**1**);**

**if(**ret **==** 0**){**

printf**(**"peer closed\n"**);**

**break;**

**}**

printf**(**"%s"**,**revbuf**);**

write**(**conn**,**revbuf**,**strlen**(**revbuf**));**

**}**

**}**

int main**(){**

int fd\_lst **=** 0**;**

assert**((**fd\_lst **=** socket**(**AF\_INET**,**SOCK\_STREAM**,**0**))** **!=** **-**1**);**

int n**;**

setsockopt**(**fd\_lst**,**SOL\_SOCKET**,**SO\_REUSEADDR**,&**n**,sizeof(**n**));**

struct sockaddr\_in addr\_srv**;**

memset**(&**addr\_srv**,**0**,sizeof(**addr\_srv**));**

addr\_srv**.**sin\_family **=** AF\_INET**;**

addr\_srv**.**sin\_port **=** htons**(**8000**);**

addr\_srv**.**sin\_addr**.**s\_addr **=** inet\_addr**(**"127.0.0.1"**);**

assert**(**bind**(**fd\_lst**,(**struct sockaddr**\*)&**addr\_srv**,sizeof(**addr\_srv**))** **!=** **-**1**);**

assert**(**listen**(**fd\_lst**,**SOMAXCONN**)** **!=** **-**1**);**

int conn **=** 0**;**

struct sockaddr\_in addr\_cli**;**

socklen\_t len **=** **sizeof(**addr\_cli**);**

**while(**1**){**

assert**((**conn **=** accept**(**fd\_lst**,(**struct sockaddr**\*)&**addr\_cli**,&**len**))** **!=** **-**1 **);**

printf**(**"clinet ip: %s, port: %d\n"**,**inet\_ntoa**(**addr\_cli**.**sin\_addr**),**ntohs**(**addr\_cli**.**sin\_port**));**

pid\_t pid**;**

assert**((**pid **=** fork**())** **!=** **-**1**);**

**if(**pid **==** 0**){**

close**(**fd\_lst**);**

do\_server**(**conn**);**

close**(**conn**);**

**break;**

**}**

**else{**

close**(**conn**);**

**continue;**

**}**

**}**

**return** 0**;**

**}**

1. 聊天程序（处理粘包问题）：
   1. 服务端：//会出现父进程死亡，子进程未退出的情况，使用信号的方式关闭子进程

#include <unistd.h>

#include <sys/socket.h>

#include <sys/types.h>

#include <netinet/in.h>

#include <arpa/inet.h>

#include <sys/prctl.h>

#include <signal.h>

#include <stdio.h>

#include <string.h>

#include <assert.h>

int static st\_sig **=** 0**;**

struct Packet**{**

int len**;**

char buf**[**1024**];**

**};**

void do\_sed**(**int conn**){**

Packet sedbuf**;**

memset**(&**sedbuf**,**0**,sizeof(**sedbuf**));**

**while(**fgets**(**sedbuf**.**buf**,sizeof(**sedbuf**.**buf**),**stdin**)** **!=** **NULL){**

int n **=** strlen**(**sedbuf**.**buf**);**

sedbuf**.**len **=** htons**(**n**);**

write**(**conn**,&**sedbuf**,**n**+**4**);**

memset**(&**sedbuf**,**0**,sizeof(**sedbuf**));**

**}**

**}**

void do\_rev**(**int conn**){**

Packet revbuf**;**

**while(**1**){**

memset**(&**revbuf**,**0**,sizeof(**revbuf**));**

int n **=** 0**;**

int ret **=** read**(**conn**,&**n**,**4**);**

assert**(**ret **!=** **-**1**);**

**if(**ret **==** 0**){**

printf**(**"peer closed\n"**);**

**break;**

**}**

revbuf**.**len **=** ntohs**(**n**);**

ret **=** read**(**conn**,&**revbuf**.**buf**,**revbuf**.**len**);**

assert**(**ret **!=** **-**1**);**

**if(**ret **==** 0**){**

printf**(**"peer closed\n"**);**

**break;**

**}**

printf**(**"%s"**,**revbuf**.**buf**);**

**}**

**}**

void handle**(**int sig**){**

**if(**sig **==** SIGHUP**){**

printf**(**"child received peer closed\n"**);**

st\_sig **=** 1**;**

**}**

**}**

int main**(){**

int fd\_lst **=** 0**;**

assert**((**fd\_lst **=** socket**(**AF\_INET**,**SOCK\_STREAM**,**0**))** **!=** **-**1**);**

int n**;**

setsockopt**(**fd\_lst**,**SOL\_SOCKET**,**SO\_REUSEADDR**,&**n**,sizeof(**n**));**

struct sockaddr\_in srv\_addr**;**

srv\_addr**.**sin\_family **=** AF\_INET**;**

srv\_addr**.**sin\_port **=** htons**(**8000**);**

srv\_addr**.**sin\_addr**.**s\_addr **=** inet\_addr**(**"127.0.0.1"**);**

assert**((**bind**(**fd\_lst**,(**struct sockaddr**\*)&**srv\_addr**,sizeof(**srv\_addr**)))** **!=** **-**1**);**

assert**((**listen**(**fd\_lst**,**SOMAXCONN**))** **!=** **-**1**);**

int conn **=** 0**;**

sockaddr\_in cli\_addr**;**

socklen\_t len **=** **sizeof(**cli\_addr**);**

assert**((**conn **=** accept**(**fd\_lst**,(**struct sockaddr**\*)&**cli\_addr**,&**len**))** **!=** **-**1**);**

printf**(**"Connect from %s : %d\n"**,**inet\_ntoa**(**cli\_addr**.**sin\_addr**),**ntohs**(**cli\_addr**.**sin\_port**));**

pid\_t pid **=** 0 **;**

assert**((**pid **=** fork**())** **!=** **-**1**);**

**if(**pid **==** 0**){**

signal**(**SIGHUP**,**handle**);**

prctl**(**PR\_SET\_PDEATHSIG**,**SIGHUP**);**

sleep**(**1**);**

**if(**st\_sig**){**

printf**(**"child closed\n"**);**

**return** 0**;**

**}**

do\_sed**(**conn**);**

**}**

**else{**

do\_rev**(**conn**);**

**}**

close**(**conn**);**

close**(**fd\_lst**);**

**return** 0**;**

**}**

* 1. 客户端：

#include <unistd.h>

#include <sys/socket.h>

#include <sys/types.h>

#include <netinet/in.h>

#include <arpa/inet.h>

#include <signal.h>

#include <sys/prctl.h>

#include <stdio.h>

#include <string.h>

#include <assert.h>

#include <stdlib.h>

struct Packet**{**

int len**;**

char buf**[**1024**];**

**};**

void do\_sed**(**int conn**){**

Packet sedbuf**;**

memset**(&**sedbuf**,**0**,sizeof(**sedbuf**));**

**while(**fgets**(**sedbuf**.**buf**,sizeof(**sedbuf**.**buf**),**stdin**)** **!=** **NULL){**

int n **=** strlen**(**sedbuf**.**buf**);**

sedbuf**.**len **=** htons**(**n**);**

write**(**conn**,&**sedbuf**,**n**+**4**);**

memset**(&**sedbuf**,**0**,sizeof(**sedbuf**));**

**}**

**}**

void do\_rev**(**int conn**){**

Packet revbuf**;**

**while(**1**){**

memset**(&**revbuf**,**0**,sizeof(**revbuf**));**

int n **=** 0**;**

int ret **=** read**(**conn**,&**n**,**4**);**

assert**(**ret **!=** **-**1**);**

**if(**ret **==** 0**){**

printf**(**"peer closed\n"**);**

**break;**

**}**

revbuf**.**len **=** ntohs**(**n**);**

ret **=** read**(**conn**,**revbuf**.**buf**,**revbuf**.**len**);**

assert**(**ret **!=** **-**1**);**

**if(**ret **==** 0**){**

printf**(**"peer closed\n"**);**

**break;**

**}**

printf**(**"%s"**,**revbuf**.**buf**);**

**}**

**}**

void handle**(**int sig**){**

**if(**sig **==** SIGHUP**){**

printf**(**"handle \n"**);**

exit**(**0**);**

**}**

**}**

int main**(){**

int sock **=** 0**;**

assert**((**sock **=** socket**(**AF\_INET**,**SOCK\_STREAM**,**0**))** **!=** **-**1**);**

struct sockaddr\_in srv\_addr**;**

srv\_addr**.**sin\_family **=** AF\_INET**;**

srv\_addr**.**sin\_port **=** htons**(**8000**);**

srv\_addr**.**sin\_addr**.**s\_addr **=** inet\_addr**(**"127.0.0.1"**);**

assert**(**connect**(**sock**,(**struct sockaddr**\*)&**srv\_addr**,sizeof(**srv\_addr**))** **!=** **-**1**);**

pid\_t pid **=** 0 **;**

assert**((**pid **=** fork**())** **!=** **-**1**);**

**if(**pid **==** 0**){**

signal**(**SIGHUP**,**handle**);**

prctl**(**PR\_SET\_PDEATHSIG**,**SIGHUP**);**

do\_sed**(**sock**);**

**}**

**else{**

do\_rev**(**sock**);**

**}**

close**(**sock**);**

**return** 0**;**

**}**

1. gethostname() gethostbyname()

#include <unistd.h>

#include <netdb.h>

#include <netinet/in.h>

#include <arpa/inet.h>

#include <stdio.h>

#include <string.h>

#include <assert.h>

int main**(){**

char name**[**1024**];**

assert**(**gethostname**(**name**,sizeof(**name**))** **!=** **-**1**);**

printf**(**"%s\n"**,**name**);**

struct hostent**\*** he**;**

he **=** gethostbyname**(**name**);**

int len **=** he**->**h\_length**;** //并不代表有几个ip，代表ip总共占用的字节数

int i **=** 0 **;**

**while(**he**->**h\_addr\_list**[**i**]** **!=** **NULL){**

printf**(**"%s\n"**,**inet\_ntoa**(\*(**struct in\_addr**\*)**he**->**h\_addr\_list**[**i**]));**//必须使用 inet\_ntoa 将网络地址转化

**++**i**;**

**}**

**return** 0**;**

**}**

1. tcp的11种状态
   1. SYN\_SEND
   2. LISTEN
   3. SYN\_RECD
   4. ESTIBLISHED
   5. FIN\_WAIT\_1
   6. CLOSED\_WAIT
   7. FIN\_WAIT\_2
   8. LAST\_ACK
   9. TIME\_WAIT
   10. CLOSED
   11. CLOSING(特殊情况下出现，当客户端和服务端同时发送FIN信号，不会出现FIN\_WAIT\_1,FIN\_WAIT\_2状态，取而代之的是CLOSING状态)
2. SIGPIPE

当对接方关闭的情况下，发送信息，对接方会恢复RST，在接收到RST后，还继续给对接方发送信息，就会收到对接方返回的SIGPIPE信号

1. select：

1 #include**<**unistd**.**h**>**

2 #include**<**sys**/**types**.**h**>**

3 #include**<**sys**/**socket**.**h**>**

4 #include**<**netinet**/**in**.**h**>**

5 #include**<**arpa**/**inet**.**h**>**

6 #include**<**sys**/**time**.**h**>**

7 #include**<**errno**.**h**>**

8 #include**<**signal**.**h**>**

9

10 #include**<**stdio**.**h**>**

11 #include**<**string**.**h**>**

12 #include**<**assert**.**h**>**

13

14 void handle**(**int sig**){**

15 printf**(**"sig %d\n"**,**sig**);**

16 **}**

17 int main**(){**

18

19 signal**(**SIGPIPE**,**handle**);**

20 int sock **=** 0**;**

21 assert**((**sock **=** socket**(**AF\_INET**,**SOCK\_STREAM**,**0**))** **!=** **-**1**);**

22 struct sockaddr\_in addr**;**

23 addr**.**sin\_family **=** AF\_INET**;**

24 addr**.**sin\_port **=** htons**(**8000**);**

25 int n **=** 0**;**

26 setsockopt**(**sock**,**SOL\_SOCKET**,**SO\_REUSEADDR**,&**n**,sizeof(**n**));**

27 addr**.**sin\_addr**.**s\_addr **=** inet\_addr**(**"127.0.0.1"**);**

28 assert**(**connect**(**sock**,(**struct sockaddr**\*)&**addr**,sizeof(**addr**))** **!=** **-**1**);**

29 char revbuf**[**1024**]** **=** **{**0**}** **;**

30 char sedbuf**[**1024**]** **=** **{**0**};**

31 fd\_set read\_set**;**

32 FD\_ZERO**(&**read\_set**);**

33 int in\_no **=** fileno**(**stdin**);**

34 int maxfd **=** sock**+**1**;**

35 **while(**1**){**

36 FD\_SET**(**in\_no**,&**read\_set**);**//注意：这两句必须写在while循环内

37 FD\_SET**(**sock**,&**read\_set**);**

38 int ret **=** select**(**maxfd**,&**read\_set**,NULL,NULL,NULL);**//在此处有数据阻塞，直到感兴趣的fd有数据可读

39 **if(**ret **<=** 0**){**

40 printf**(**"select error\n"**);**

41 **return** 0**;**

42 **}**

43 **if(**FD\_ISSET**(**sock**,&**read\_set**)){**

44 int ret **=** read**(**sock**,**revbuf**,sizeof(**revbuf**));**

45 **if(**ret **==** **-**1 **&&** errno **==** EINTR**){**

46 **continue;**

47 **}**

48 **if(**ret **==** **-**1**){**

49 printf**(**"error\n"**);**

50 **break;**

51 **}**

52 **if(**ret **==** 0**){**

53 printf**(**"peer closed\n"**);**

54 char s**[**1024**]** **=** "11111"**;**

55 write**(**sock**,**s**,**strlen**(**s**));**//当对等方关闭，sock会不阻塞，read返回0，该句执行后，srv回应RST

56 write**(**sock**,**s**,**strlen**(**s**));**//收到RST后，继续发送数据，收到SIGPIPE

57 **break;**

58 **}**

59 printf**(**"%s"**,**revbuf**);**

60 **}**

61 **if(**FD\_ISSET**(**in\_no**,&**read\_set**)){**

62 fgets**(**sedbuf**,sizeof(**sedbuf**),**stdin**);**

63 write**(**sock**,**sedbuf**,**strlen**(**sedbuf**));**

64 **}**

65 memset**(**sedbuf**,**0**,sizeof(**sedbuf**));**

66 memset**(**revbuf**,**0**,sizeof(**revbuf**));**

67 **}**

68 close**(**sock**);**

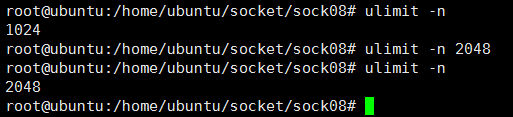
69 **return** 0**;**

70 **}**

1. int shutdown(int sockfd, int how);成功返回0，失败返回-1；how取值：SHUT\_RD,SHUT\_WR,SHUT\_RDWR
   1. 可以选择关闭写或者读或者读写
   2. 和引用计数无关，close只是引用计数减一，为0时关闭
2. select缺陷和poll的使用
   1. select缺点：
      1. 并发数受进程能打开最大文件数目限制

使用ulimit –u查询进程最大打开文件数

ulimit –u N改变当前进程最大打开文件数



通过程序改变进程所能打开的最大文件数：

27 /\*

28 getrlimit(int resource,struct rlimit\* rlim);

29 setrlimit(int resource,const struct limit\* rlim)

30 struct rlimit{

31 rlim\_t rlim\_cur;

32 rlim\_t rlim\_max;

33 }

34 resource = RLIMIT\_NOFILE

35 \*/

36 struct rlimit rlim**;**

37 assert**((**getrlimit**(**RLIMIT\_NOFILE**,&**rlim**))!=-**1**);**

38 printf**(**"%d, %d\n"**,(**int**)**rlim**.**rlim\_cur**,(**int**)**rlim**.**rlim\_max**);**

39 rlim**.**rlim\_cur **=** 3072**;**

40 rlim**.**rlim\_max **=** 8088**;**

41 assert**((**setrlimit**(**RLIMIT\_NOFILE**,&**rlim**))** **!=** **-**1**);**

42 printf**(**"%d, %d\n"**,(**int**)**rlim**.**rlim\_cur**,(**int**)**rlim**.**rlim\_max**);**

**输出：**

**1024，2048**

**3072，8088**

* + 1. 并发数受fd\_set容量限制，FD\_SETSIZE

只能通过编译内核改变，宏变量

实验：已将进程所能打开的最大文件数改为了>1024

结果：服务端能处理的连接数数1020（因为0（stdin），1（stdout），2（stderr），3（fd\_lst）已经打开）；客户端能连接1021（较服务端少监听fd\_lst）

* 1. poll使用

服务端：

27 int count **=** 0**;**

28 struct pollfd fds**[**2048**]** **=** **{**0**};**

29 **for(**int i **=** 0 **;** i **<** 2048**;++**i**){**

30 fds**[**i**].**fd **=** **-**1**;**

31 **}**

32 fds**[**0**].**fd **=** fd\_lst**;**

33 fds**[**0**].**events **=** POLLIN **;**

34 int maxi **=** 1**;**

35 **while(**1**){**

36 assert**((**poll**(**fds**,**maxi**,-**1**))>**0**);**

37 memset**(&**cliaddr**,**0**,sizeof(**cliaddr**));**

38 assert**((**conn **=** accept**(**fd\_lst**,(**struct sockaddr**\*)&**cliaddr**,&**len**))** **!=** **-**1**);**

39 printf**(**"count = %d , addr = %s , port = %d\n"**,++**count**,**inet\_ntoa**(**cliaddr**.**sin\_addr**),**ntohs**(**cliaddr**.**sin\_port**)** **);**

40 **}**

最大可连接数大于1024