0216socket_2

TcpServer

```
SOCKET(2)                                  Linux Programmer's Manual

NAME
       socket - create an endpoint for communication

SYNOPSIS
       #include <sys/types.h>          /* See NOTES */
       #include <sys/socket.h>

       int socket(int domain, int type, int protocol);

DESCRIPTION
       socket() creates an endpoint for communication and returns a descriptor.

       The  domain argument specifies a communication domain; this selects the protocol family

       AF_ATMPVC          Access to raw ATM PVCs
       AF_APPLETALK       Appletalk                    ddp(7)
       AF_PACKET          Low level packet interface   packet(7)

       The socket has the indicated type, which specifies the communication semantics.  Currently defined types are:

       SOCK_STREAM     Provides sequenced, reliable, two-way, connection-based byte  streams.  An  out-of-band  data
                       transmission mechanism may be supported.

       SOCK_DGRAM      Supports datagrams (connectionless, unreliable messages of a fixed maximum length).

       SOCK_SEQPACKET  Provides  a sequenced, reliable, two-way connection-based data transmission path for datagrams
                       of fixed maximum length; a consumer is required to read an entire packet with each input  sys-
                       tem call
```

tcp

面向字节流的，可靠的数据传输

不可靠的数据传输

udp

1. 创建套接字
2. bind绑定

前两步和udp是一样的，但是tcp是面向连接的！所以tcp有第三步建立连接，即握手！

如果一个服务器需要被连接
那么他就必须出以一种：等待连接的状态，即监听状态

```
LISTEN(2)                  Linux Programmer's Manual                 LISTEN(2)

NAME
       listen - listen for connections on a socket

SYNOPSIS
       #include <sys/types.h>          /* See NOTES */
       #include <sys/socket.h>

       int listen(int sockfd, int backlog);

DESCRIPTION
       listen() marks the socket referred to by sockfd as a passive socket, that is, as a socket that will be used to
       accept incoming connection requests using accept(2).

       The sockfd argument is a file descriptor that refers to a socket of type SOCK_STREAM or SOCK_SEQPACKET.

       The backlog argument defines the maximum length to which the queue of pending connections for sockfd may grow.
       If  a connection request arrives when the queue is full, the client may receive an error with an indication of
       ECONNREFUSED or, if the underlying protocol supports retransmission, the request may  be  ignored  so  that  a
       later reattempt at connection succeeds.
```

```
-rw-rw-r-- 1 yufc yufc  1741 Apr 14 17:19 tcp_server.hpp
(base) [yufc@VM-12-12-centos:~/Core/BitCodeField/0215/tcp]$ netstatt -antp
bash: netstatt: command not found
(base) [yufc@VM-12-12-centos:~/Core/BitCodeField/0215/tcp]$ ./tcp_server 8080
[NORMAL] [1681464079] create socket success, sock: 3
[NORMAL] [1681464079] init TcpServer Success
```

第一个参数就是套接字

第二个参数以我们目前的知识储备暂时还无法理解

```
(base) [yufc@VM-12-12-centos:~/Core]$ netstat -antp
(Not all processes could be identified, non-owned process info
 will not be shown, you would have to be root to see it all.)
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 0.0.0.0:8080            0.0.0.0:*               LISTEN      9834/./tcp_server
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN      -
tcp        0      0 127.0.0.1:25            0.0.0.0:*               LISTEN      -
tcp        0      0 127.0.0.1:33420         0.0.0.0:*               LISTEN      29416/node
tcp        0      0 127.0.0.1:47688         127.0.0.1:33420         ESTABLISHED -
```

此时通过：
netstat –antp 查看当前机子的所有tcp服务

---

4. 获取连接

第二个参数是输出型参数
第三个参数是输入输出型参数

和udp的recvfrom的后面两个参数一模一样

```
NAME
       accept, accept4 - accept a connection on a socket

SYNOPSIS
       #include <sys/types.h>          /* See NOTES */
       #include <sys/socket.h>

       int accept(int sockfd, struct sockaddr *addr, socklen_t *addrlen);

       #define _GNU_SOURCE             /* See feature_test_macros(7) */
       #include <sys/socket.h>

       int accept4(int sockfd, struct sockaddr *addr,
                   socklen_t *addrlen, int flags);

DESCRIPTION
       The  accept()  system  call  is  used  with  connection-based socket types (SOCK_STREAM,
       SOCK_SEQPACKET).  It extracts the first connection request on the queue of pending  con-
       nections for the listening socket, sockfd, creates a new connected socket, and returns a
       new file descriptor referring to that socket.  The newly created socket is  not  in  the
       listening state.  The original socket sockfd is unaffected by this call.
```

```
void start()
{
    while (true)
    {
        // 4. 获取连接
        struct sockaddr_in src;
        socklen_t len = sizeof(src);
        int sock = accept(__sock, (sockaddr *)&src, &len);
        // 它的返回值也是一个套接字
    }
}
```

两个套接字有什么区别？

如果忘记了就看代码
看视频

写完第一版的服务端之后，时间不够
不足以支撑我们写完客户端
但是这里介绍一个工具，可以暂时代替客户端

```
(base) [yufc@VM-12-12-centos:~/Core/BitCodeField/0215/tcp]$ telnet 127.0.0.1 8080
Trying 127.0.0.1...
Connected to 127.0.0.1.
Escape character is '^]'.
```

./tcp_server
telnet tcp

用telnet这个工具就行

我们创建两个客户端同时对服务端进行测试，
我们发现，我们写的这个version1，一次只能处理一个客户端
处理完一个才能处理下一个，很明显，这个是不能够被使用的！
为什么？如何解决？

```
127.0.0.1: 51918 # nihao

127.0.0.1: 51918 # zaima

127.0.0.1: 51918 # ????
127.0.0.1: 51918 # ??
127.0.0.1: 51918 # ????
127.0.0.1: 51918 # ??
127.0.0.1: 51918 #

127.0.0.1: 51918 # ????
127.0.0.1: 51918 # ??
127.0.0.1: 51918 # q

127.0.0.1: 51918 # quit

127.0.0.1: 51918 # exit

[NORMAL] [1681525738] 127.0.0.1: 51918 shutdown, me too!
```

版本二：多进程版

```
    // 创建子进程 -- 让子进程给新的连接提供服务，子进程能不能打开父进程曾经打开的fd呢?
    pid_t id = fork();
    if(id == 0)
    {
        // 子进程 -- 我们可以让子进程进行读取，父进程继续提供上面的服务
        // 但是如果子进程处理完之后，退出后，会进入僵尸状态
        exit(0);
    }
    // 父进程
    // waitpid(); // 阻塞式等待
    close(service_sock);
}

}
void start()
{
    signal(SIGCHLD, SIG_IGN); // 对于SIGCHLD，主动忽略SIGCHLD信号，子进程退出之后
                              // 会自动释放，父进程等都不用等
    while (true)
    {
        // 4. 获取连接
        struct sockaddr_in src;
        socklen_t len = sizeof(src);
        int service_sock = accept(__listen_sock, (sockaddr *)&src, &len);
        // 它的返回值也是一个套接字
```

如何处理子进程退出后的僵尸状态？

?

首先当然不能阻塞式等待
也不能轮询检测，因为轮询检测太麻烦了，
还要保存所有子进程pid

```
(base) [yufc@VM-12-12-centos:~/Core]$ telnet 127.0.0.1 8080
Trying 127.0.0.1...
Connected to 127.0.0.1.
Escape character is '^]'.
^]

telnet>
woshidierge
woshidierge
nizaima
nizaima
```

```
(base) [yufc@VM-12-12-centos:~/Core/BitCodeField/0215/tcp]$ telnet 127.0.0.1 8080
Trying 127.0.0.1...
Connected to 127.0.0.1.
Escape character is '^]'.
^]

telnet>
nihao
nihao
zaima
zaima
```

```
(base) [yufc@VM-12-12-centos:~/Core/BitCodeField/0215/tcp]$ make clean;make
rm -f tcp_client tcp_server
g++ -o tcp_client tcp_client.cc -std=c++11 -g -lpthread
g++ -o tcp_server tcp_server.cc -std=c++11 -g -lpthread
(base) [yufc@VM-12-12-centos:~/Core/BitCodeField/0215/tcp]$ ./tcp_server  8080
[NORMAL] [1681526608] create socket success, sock: 3
[NORMAL] [1681526608] init TcpServer Success
[NORMAL] [1681526610] link success, serviceSock: 4 | 127.0.0.1 : 52858

127.0.0.1: 52858 # nihao

127.0.0.1: 52858 # zaima

[NORMAL] [1681526624] link success, serviceSock: 4 | 127.0.0.1 : 52874

127.0.0.1: 52874 # woshidierge

127.0.0.1: 52874 # nizaima
```

有多少个客户端
就有多少个子进程