设置servers为抽象类是为了接口规范。Servers is set as an abstract class for the interface specification.

**Server.h**

Class Server:

/\*initialize the servers,including setting up

        sockets,bingding port and listening.

        初始化服务器，建立套接字，绑定端口并监听

    \*/

    virtual \_\_int32\_t startUp(\_\_int32\_t\* port) = 0;

    /\*return error request返回错误请求\*/

    virtual \_\_int32\_t errorRequest(\_\_int32\_t) = 0;

    //accept http request接收http请求

    virtual void acceptRequest();

**httpd.h**

**class Httpd**

/\* This function starts the process of listening for web connections

 \* on a specified port.  If the port is 0, then dynamically allocate a

 \* port and modify the original port variable to reflect the actual

 \* port.

 \* Parameters: pointer to variable containing the port to connect on

 \* Returns: the socket

初始化服务器，建立套接字，绑定端口并监听

\*/

\_\_int32\_t Httpd::startUp(\_\_int16\_t\* port)

/\* Print out an error message with perror() (for system errors; based

 \* on value of errno, which indicates system call errors) and exit the

 \* program indicating an error.

使用perror返回一个错误信息，并告诉你错误在哪里。\*/

virtual void errorInfo (const string Info);

/\* A request has caused a call to accept() on the server port to

 \* return.  Process the request appropriately.

 \* Parameters: the socket connected to the client

 \* 返回从套接字接收到的HTTP请求 \*/

virtual void acceptRequest(\_\_int64\_t\* arg);

Get a line from a socket, whether the line ends in a newline,

 \* carriage return, or a CRLF combination.  Terminates the string read

 \* with a null character.  If no newline indicator is found before the

 \* end of the buffer, the string is terminated with a null.  If any of

 \* the above three line terminators is read, the last character of the

 \* string will be a linefeed and the string will be terminated with a

 \* null character.

 \* Parameters: the socket descriptor

 \*             the buffer to save the data in

 \*             the size of the buffer

 \* Returns: the number of bytes stored (excluding null)

从套接字中获取一行，无论该行是否以换行符结束，

\*回车，或CRLF组合。终止字符串读取

\*带有空字符。如果在之前没有发现换行指示器

\*缓冲区的结束，字符串以null结束。如果任何

的最后一个字符将读取上述三个行终止符

\*字符串将是换行符，字符串将以

\* null字符。

参数: soct,套接字描述符

Buf, 保存数据的缓冲区

Size,缓冲区的大小

\*返回:存储的字节数(不包括null)\*/

virtual \_\_int32\_t get\_line(\_\_int32\_t sock, std::string\* buf, \_\_int32\_t size);

**Basic knowledge所用到的基础知识**

/\* socket (int \_\_domain, int \_\_type, int \_\_protocol);

        param: domin tells the system which underlying  protocol family to use.

         For the TCP/IP protocol family, the parameter should be PF\_INET or PF\_INET6.

         domin 告诉系统使用哪一个底层协议族。对于tcp/ip协议族而言，该参数应为PF\_INET或者PF\_INET6。

        PF\_INET表示ipv4,PF\_INET6表示ipv6.

       The type parameter specifies the service type. The service types are SOCK\_STREAM service (stream service) and SOCK\_UGRAM (datagram) service. The former uses TCP at the transport layer, and the latter uses UDP.

type参数指定服务类型。服务类型有SOCK\_STREAM服务（流服务）和SOCK\_UGRAM（数据报）服务。前者为传输层使用tcp协议，后者为使用udp协议.

Protocol usually use 0;

socket函数调用成功返回一个socket文件描述符，失败返回-1并设置errno.

\*/

该例子出现在httpd.cpp中的第13行代码中。This example appears in line 13 of httpd.cpp.

    httpd = socket(PF\_INET, SOCK\_STREAM, 0);

**函数原型**

**设置文件描述符函数setsockopt**

int setsockopt (int \_\_fd, int \_\_level, int \_\_optname,

             const void \*\_\_optval, socklen\_t \_\_optlen)

fd形参是指定被操作的目标socket，level参数指定要操作哪个协议的选项，例如ipv4,ipv6,tcp等。Optname指定选项的名字。Val和len是被操作选项的值和长度。

函数成功调用时返回0，失败为-1并设置errno.

The fd parameter specifies the target socket to operate on, and the level parameter specifies the protocol option to operate on, such as ipv4,ipv6, TCP, and so on. Optname Specifies the name of the option. Val and len are the values and lengths of the options being manipulated.

The function returns 0 on success, -1 on failure and sets errno.

具体可看<https://docs.microsoft.com/en-us/windows/win32/api/winsock/nf-winsock-setsockopt>

该文档是微软的文档。

该例子出现在httpd.cpp中的第23行代码中。

This example appears in line 23 of httpd.cpp.

Param： httpd,要操作的sock文件描述符。HTTPD, the SOCK file descriptor to operate on.

SOL\_SOCKET，表示通用sockte选项，与协议无关。SOL\_SOCKET: indicates the general SOCKTE option, independent of the protocol.

SO\_REUSEADDR, which enforces the use of the socket address occupied by connections in TIME\_WAIT state. That is, the closed socket is quickly reclaimed so that TCP connections are not in TIME\_WAIT state at all, allowing applications to immediately reuse the local socket address.

SO\_REUSEADDR，设置该项可以强制使用处于TIME\_WAIT状态的连接占用的socket地址，即快速回收被关闭的socket，从而使得tcp连接根本就不进入TIME\_WAIT状态，进而允许应用程序立即重用本地的socket地址。

On,为被操作选项的值.

On is the value of the option being operated On.

Sizeof(on)为被操作选项的长度。

Sizeof(on) is the length of the option being operated on.

if((setsockopt(httpd, SOL\_SOCKET, SO\_REUSEADDR, &on, sizeof(on))) < 0)

        errorInfo("setsocket failed");

**读取socket文件描述符函数**

**Getsocketopt**

**如上述一致**