课程地址：【新】Linux C/C++高级全栈开发（后端/游戏/嵌入式/高性能网络/存储/基础架构） (0voice.com)

课程内容安排

2. **文件传输和接口设计，主要内容：**

- 数据库设计
- 图床接口分析
- reactor网络模型分析
- 代码实现注册功能
- 代码实现登录功能
- 代码实现我的文件列表

课程源码目录：tuchuang/tc-mini2

1 数据库设计

1.1 用户注册和登录

1.1.1 用户信息表 (user_info)

字段名	中文含义	类型	是否允许空	默认值	其他
id	序号	bigint	否		自动递增，主键
user_name	用户名称	varchar(32)	否	''	唯一
nick_name	用户昵称	varchar(32)	否	''	
phone	手机号码	varchar(16)	否	''	
email	邮箱	varchar(64)	否	''	
password	密码	varchar(32)	是	''	保存md5后的加密值
create_time	用户创建时间	TIMESTAMP	否	CURRENT_TIMESTAMP	

1.2 我的文件列表

1.2.1 用户文件列表 (user_file_list)

字段名	中文含义	类型	是否允许空	默认值	其他
id	序号	int	否		自动递增，主键
user	文件所属用户	varchar(32)	否		
md5	文件md5	varchar(256)	否		
file_name	文件名字	varchar(128)	否	“	
shared_status	共享状态	int	否	0	0为没有共享， 1为共享
pv	文件下载量	int	否	0	默认值为0，下载一次加1
create_time	文件创建时间	timestamp	否	CURRENT_TIMESTAMP	

1.2.2 文件信息表 (file_info)

字段名	中文含义	类型	是否允许空	默认值	其他
id	序号	bigint	否		自动递增，主键
md5	文件md5	varchar(256)	否	''	
file_id	文件id	varchar(256)	否	''	格式例如：/group1/M00/00/00/xxx.png
url	文件url	varchar(512)	否	''	格式例如：192.168.1.2:80/group1/M00/00/00/xxx.png
size	文件大小	bigint	否	0	以字节为单位
type	文件类型	varchar(32)	是	''	png, zip, mp4.....

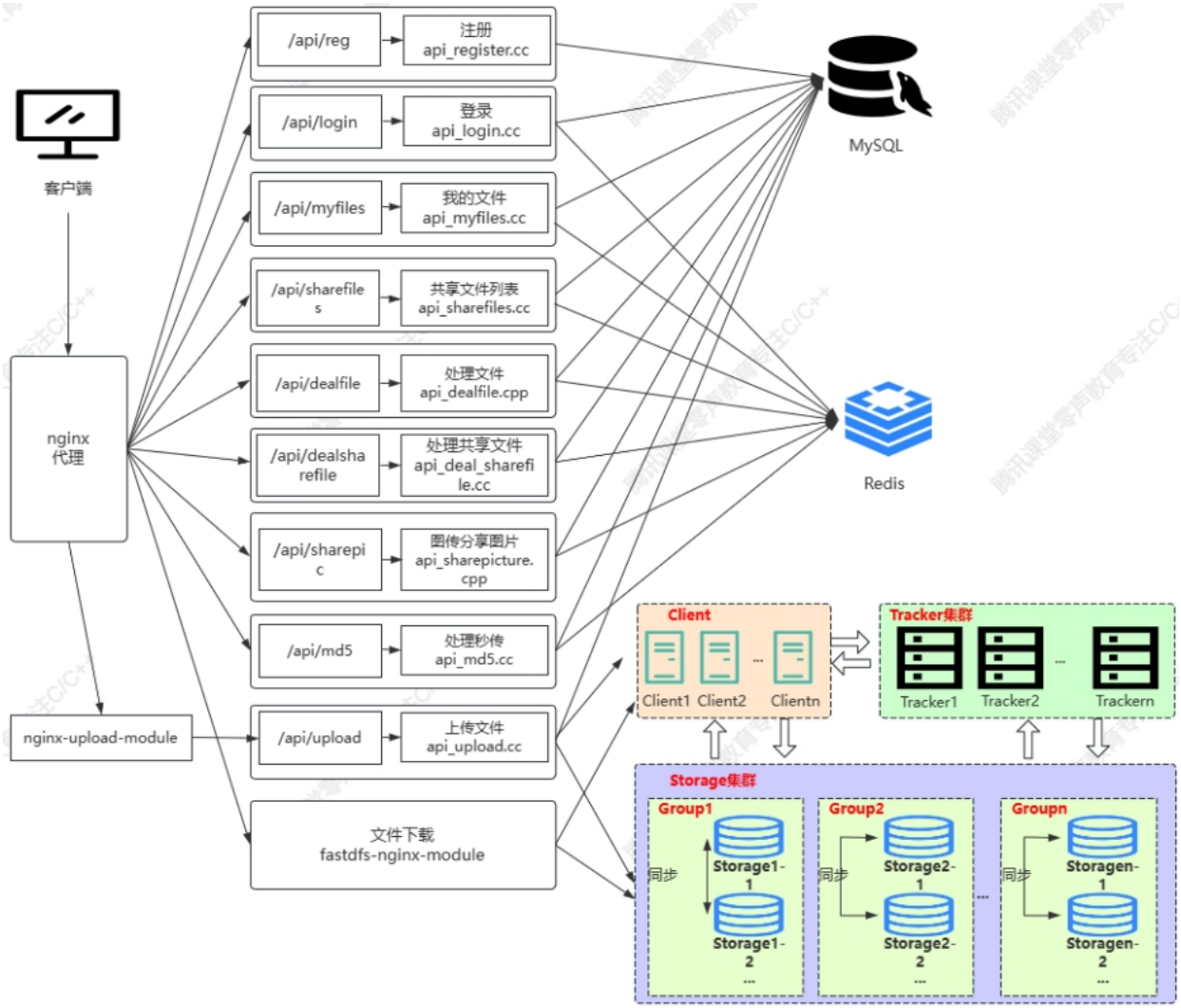
字段名	中文含义	类型	是否允许空	默认值	其他
count	文件引用计数	int	否	0	默认为1, 每增加一个用户拥有此文件，此计数器+1

2 reactor+http框架

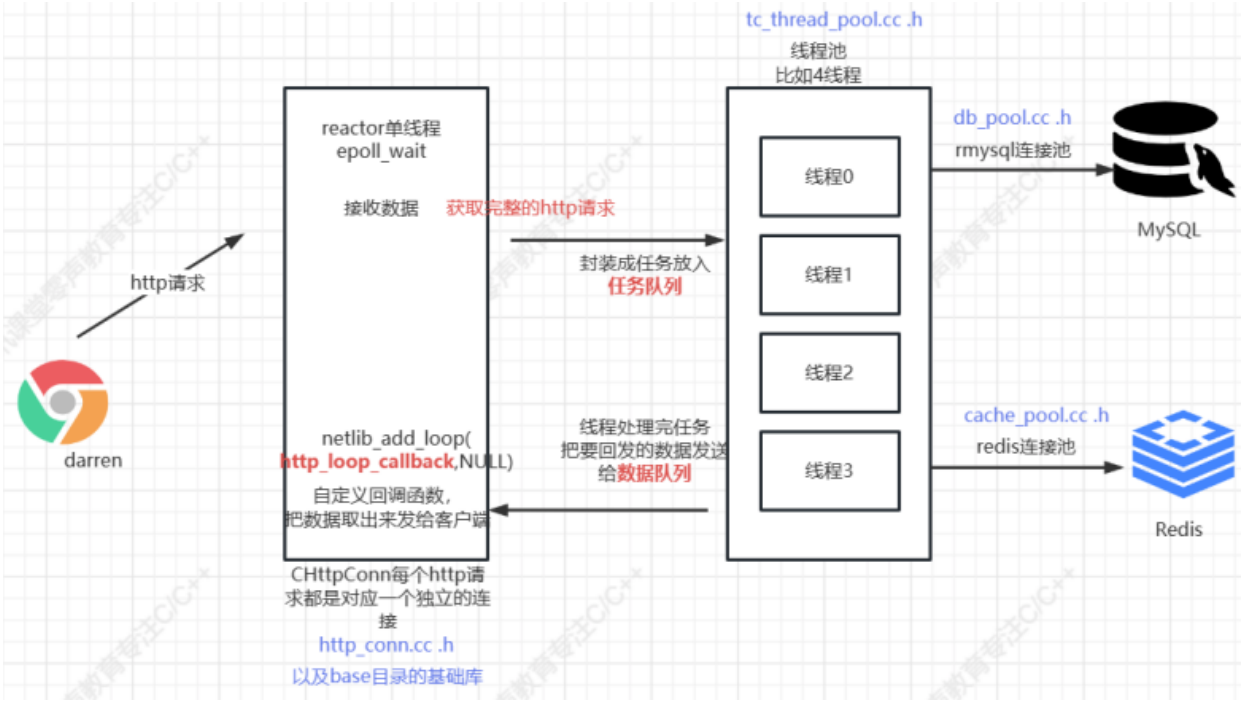
2.0 在线http请求测试

1. http在线测试：https://getman.cn/
2. http本地工具：使用**Apifox工具**更合适，下载地址：Apifox - API 文档、调试、Mock、测试一体化协作平台。拥有接口文档管理、接口调试、Mock、自动化测试等功能，接口开发、测试、联调效率，提升 10 倍。最好用的接口文档管理工具，接口自动化测试工具。
    - 比postman更好用
3. Md5在线生成：https://www.cmd5.com/
4. Json在线校验：https://www.bejson.com/json/format/

2.1 后台数据处理框架



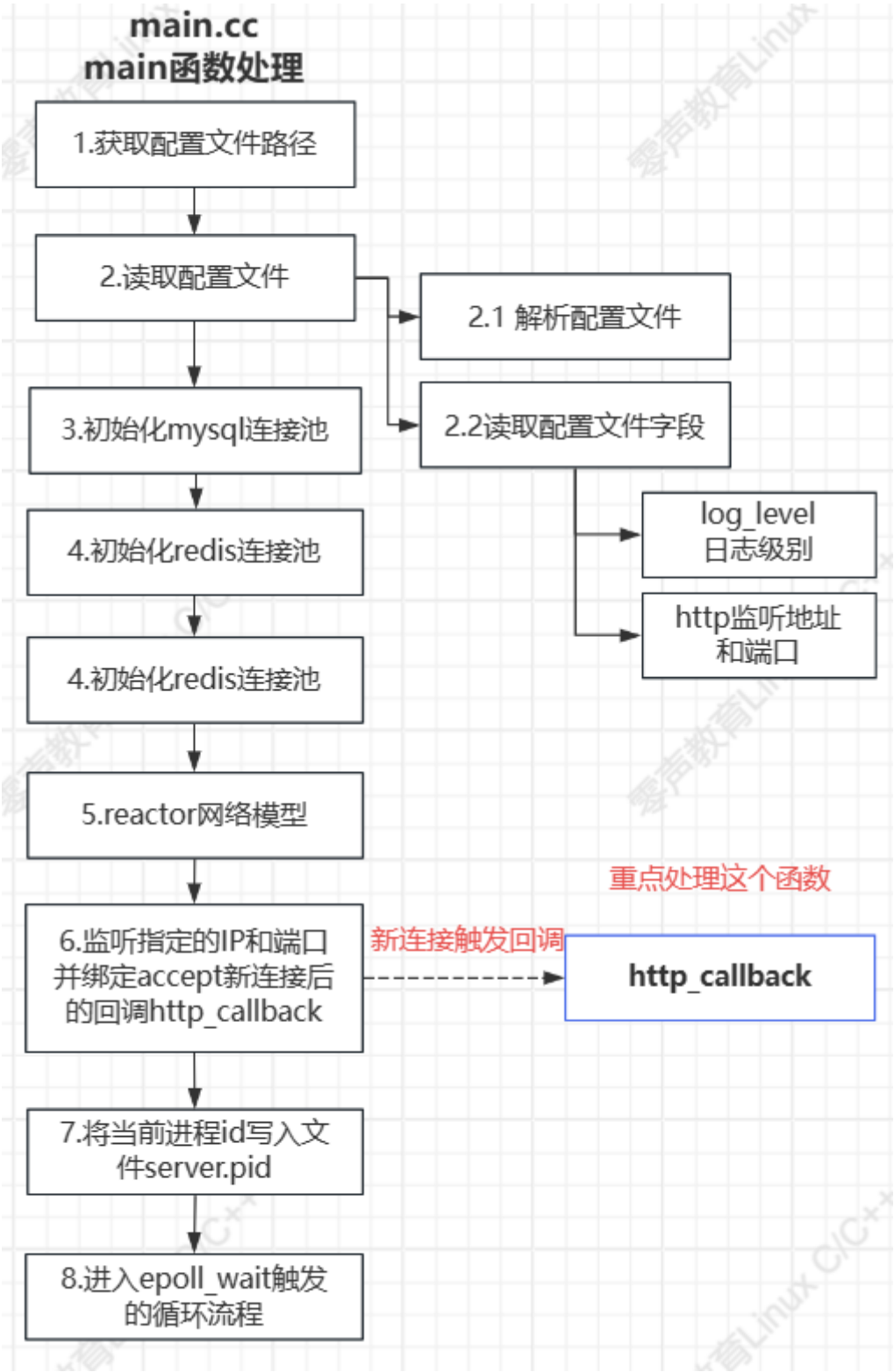
2.2 reactor网络模型



这节课先不搞那么复杂，我们先不用线程池，即是这节课是在epoll_wait所在线程处理http业务。

2.3 http业务构建

2.3.1 main函数处理流程（main.cc）



2.3.2 http请求封装和处理

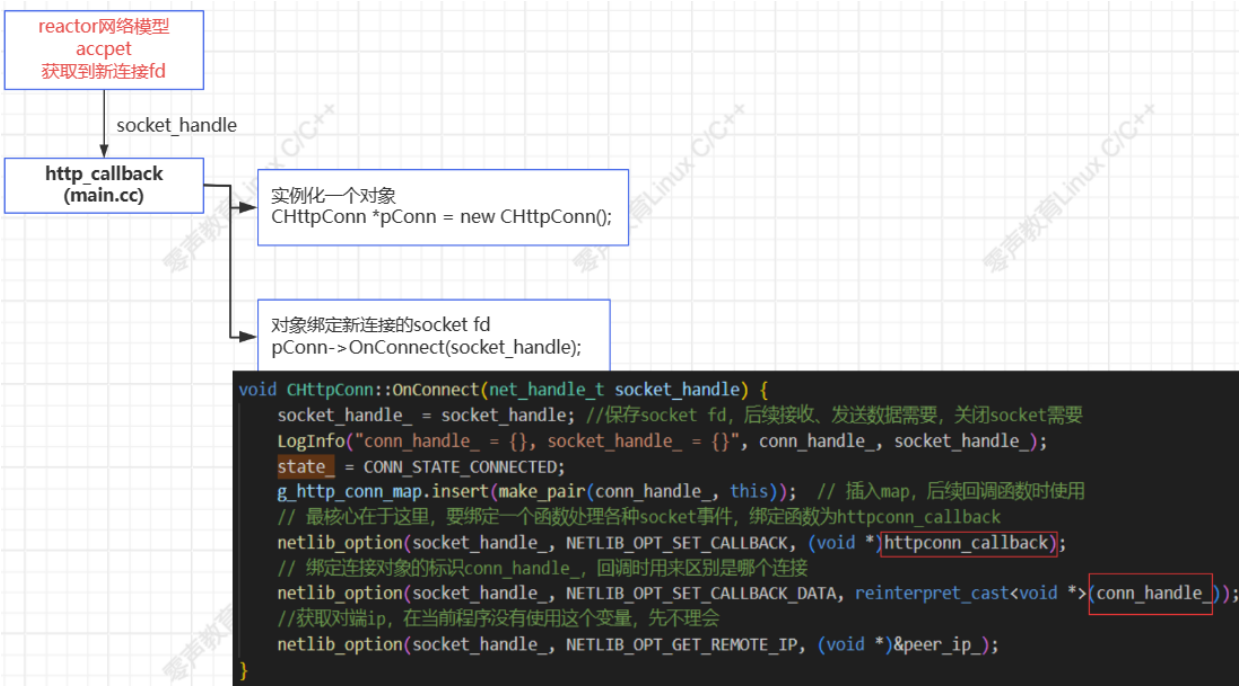
http对象封装

```
// 继承引用计数的基类，当引用计数减到0时析构对象
class CHttpConn : public CRefObject {
public:
    //构造函数
    CHttpConn();
    //析构函数
    virtual ~CHttpConn();
    // 获取当前自定义的唯一handle
    uint32_t GetConnHandle() { return conn_handle_; }
    // 返回对端ip
    char *GetPeerIP() { return (char *)peer_ip_.c_str(); }
    //发送数据，最终调用socket的send
    int Send(void *data, int len);
    //关闭连接，最终调用socket的close
    void Close();
    // accept收到新连接fd，触发OnConnect的调用，并把socket fd传递进来
    void OnConnect(net_handle_t socket_handle);
    // 可读事件触发回调OnRead
    void OnRead();
    // 可写事件触发回调OnWrite
    void OnWrite();
    // 关闭事件触发OnClose
    void OnClose();
    // 数据发送完毕回调OnWriteComlete，自己业务处理的
    void OnWriteComlete();

private:
    // 账号注册处理
    void _HandleRegisterRequest(string &url, string &post_data);
    // 账号登陆处理
    void _HandleLoginRequest(string &url, string &post_data);
    //获取我的文件列表
    void _HandleMyfilesRequest(string &url, string &post_data);

protected:
    //socket fd
    net_handle_t socket_handle_;
    // 业务自定义唯一标识
    uint32_t conn_handle_;
    //当前socket写缓存是否有空间，=ture当前socket写缓存已满，=false当前socket写缓存有空间可以
写
    bool busy_;
    //连接状态
    uint32_t state_;
    //保存对端ip
    std::string peer_ip_;
    //保存对端端口
```

```
    uint16_t peer_port_;
    //缓存从socket读取的数据
    CSimpleBuffer in_buf_;
    //缓存还没有发送出去的数据
    CSimpleBuffer out_buf_;

    // http解析
    CHttpParserWrapper http_parser_;
};
```

http连接对象初始化和响应函数

首先我们要明白reactor网络模型核心封装之一：一个新的连接 一般都会实例化一个对象绑定连接的fd，这里我们类设计为CHttpConn，所以每个新连接进来后我们要实例化一个CHttpConn对象绑定fd。

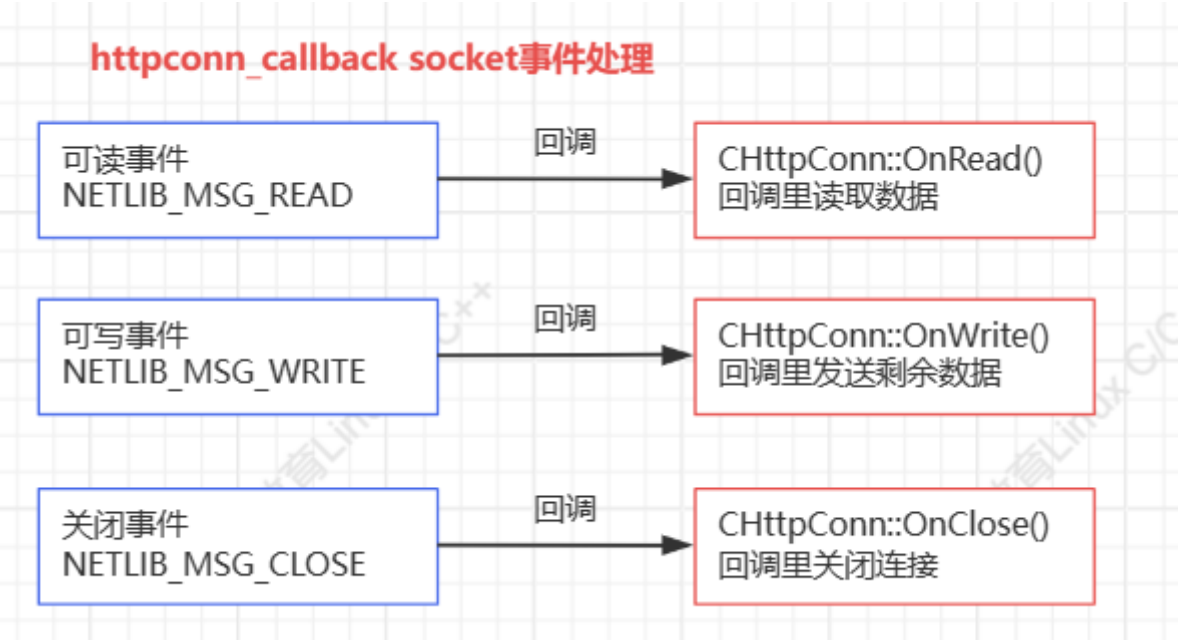


后续有可读、可写、或者关闭事件的回调都触发httpconn_callback这个函数，通过conn_handle_识别具体是哪个连接

```
// 连接的处理函数
void httpconn_callback(void *callback_data, uint8_t msg, uint32_t handle,
                       uint32_t uParam, void *pParam) {
    NOTUSED_ARG(uParam);
    NOTUSED_ARG(pParam);

    // convert void* to uint32_t, oops
    uint32_t conn_handle = *((uint32_t *)(&callback_data)); //获取连接标识
    CHttpConn *pConn = FindHttpConnByHandle(conn_handle);   //根据连接标识找到对应的http对象
```

```
    if (!pConn) {    //如果没有则返回
        LogWarn("no find conn_handle: {}", conn_handle);
        return;
    }
    pConn->AddRef();         //添加引用计数
    switch (msg) {
    case NETLIB_MSG_READ:    //可读事件
        pConn->OnRead();
        break;
    case NETLIB_MSG_WRITE:   //可写事件
        pConn->OnWrite();
        break;
    case NETLIB_MSG_CLOSE:
        pConn->OnClose();    //关闭事件
        break;
    default:
        LogError("!!!httpconn_callback error msg:{}", msg);
        break;
    }
    pConn->ReleaseRef(); //释放引用计数，如果数据发送完毕，对应的http 连接在这个位置为析构对象
}
```

即是



核心在于CHttpConn::OnRead()的处理。

数据发送完毕后触发CHttpConn::OnWriteComlete()，然后在该回调里关闭连接。

2.3.3 CHttpConn::OnRead() 请求数据处理

1. 该怎么处理？先确保能读取到数据，然后把数据打印出来观察。
2. 然后再通过http解析模块进行解析
3. 解析出对应的url后再调用对应的函数处理

```
void CHttpConn::OnRead() // CHttpConn业务层面的OnRead
{
    LogInfo("conn_handle_ = {}, socket_handle_ = {}", conn_handle_, socket_handle_);
    // 1. 把能读取的数据都读取出来
    for (;;) {
        uint32_t free_buf_len = in_buf_.GetAllocSize() - in_buf_.GetWriteOffset();
        if (free_buf_len < READ_BUF_SIZE + 1)   //这里多预留一个字节的目的是加上结束符时不会越界
            in_buf_.Extend(READ_BUF_SIZE + 1);
        //读取socket数据
        int ret = netlib_recv(socket_handle_,
                              in_buf_.GetBuffer() + in_buf_.GetWriteOffset(),
                              READ_BUF_SIZE);
        if (ret <= 0)
            break; //没有数据可以读取了

        in_buf_.IncWriteOffset(ret); // 更新下一个接收数据的位置
    }

    // 2.通过http模块解析http请求的数据
    char *in_buf = (char *)in_buf_.GetBuffer();
    uint32_t buf_len = in_buf_.GetWriteOffset();
    in_buf[buf_len] = '\0'; // 末尾加上结束符 方便分析结束位置和打印，这里之所以不越界是因为有预留in_buf_.Extend(READ_BUF_SIZE + 1)
    // 如果buf_len 过长可能是受到攻击，则断开连接 ，目前我们接受的所有数据长度不得大于2K
    if (buf_len > 2048) {
        LogError("get too much data: {}", in_buf);
        Close();
        return;
    }
    LogDebug("buf_len: {}, in_buf: {}", buf_len, in_buf); //将请求的数据都打印出来，方便调试分析http请求
    // 解析http数据
    http_parser_.ParseHttpContent(in_buf, buf_len); // 1. 从socket接口读取数据；2.然后把数据放到buffer in_buf; 3.http解析
    if (http_parser_.IsReadAll()) {
```

```
        string url = http_parser_.GetUrl();
        string content = http_parser_.GetBodyContent();
        LogInfo("url: {}", url);                    // for debug
        // 根据url处理不同的业务
        if (strncmp(url.c_str(), "/api/reg", 8) == 0) { // 注册  url 路由。 根据根据url
快速找到对应的处理函数， 能不能使用map，hash
            _HandleRegisterRequest(url, content);
        } else if (strncmp(url.c_str(), "/api/login", 10) == 0) { // 登录
            _HandleLoginRequest(url, content);
        } else if (strncmp(url.c_str(), "/api/myfiles", 10) == 0) { //获取我的文件数量
            _HandleMyfilesRequest(url, content);
        }  else {
            LogError("url unknown, url= {}", url);
            Close();
        }
    }
}
```

3 具体业务逻辑

创建一个新的数据库

mysql> create database tuchuang_mini;

mysql> use tuchuang_mini;

后续记得tc_http_server.conf数据库配置为 tuchuang_mini；

3.1 注册/api/reg api_register.cc

开发逻辑

1. **定义用户信息表，重点字段**

字段名	中文含义	类型	是否允许空	默认值	其他
id	序号	bigint	否		自动递增，主键

字段名	中文含义	类型	是否允许空	默认值	其他
user_name	用户名称	varchar(32)	否	''	唯一
nick_name	用户昵称	varchar(32)	否	''	
phone	手机号码	varchar(16)	否	''	
email	邮箱	varchar(64)	是	''	
password	密码	varchar(32)	否	''	保存md5后的加密值
create_time	用户创建时间	TIMESTAMP	否	CURRENT_TIMESTAMP	

用户名称和用户昵称都是唯一的，必填字段：

1. 用户名称
2. 用户名昵称
3. 用户密码

注册前先查询用户名是否存在，如果存在则返回错误。不存在则继续注册。

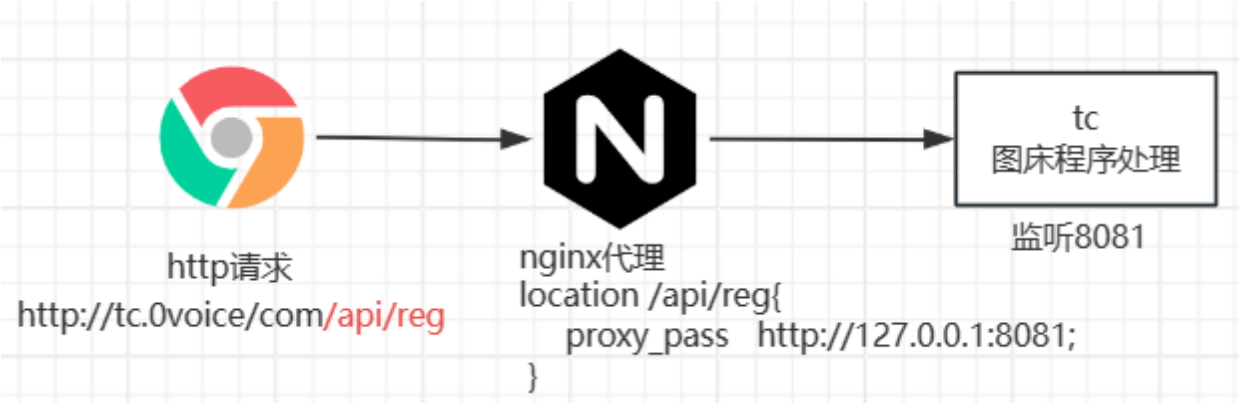
创建命令：

```
DROP TABLE IF EXISTS `user_info`;
CREATE TABLE `user_info` (
  `id` bigint(20) NOT NULL AUTO_INCREMENT COMMENT '用户序号，自动递增，主键',
  `user_name` varchar(32) NOT NULL DEFAULT '' COMMENT '用户名称',
  `nick_name` varchar(32) CHARACTER SET utf8mb4 NOT NULL DEFAULT '' COMMENT '用户昵称',
  `password` varchar(32) NOT NULL DEFAULT '' COMMENT '密码',
  `phone` varchar(16) NOT NULL DEFAULT '' COMMENT '手机号码',
  `email` varchar(64) DEFAULT '' COMMENT '邮箱',
  `create_time` timestamp NULL DEFAULT CURRENT_TIMESTAMP COMMENT '时间',
  PRIMARY KEY (`id`),
  UNIQUE KEY `uq_nick_name` (`nick_name`),
  UNIQUE KEY `uq_user_name` (`user_name`)
) ENGINE=InnoDB AUTO_INCREMENT=14 DEFAULT CHARSET=utf8 COMMENT='用户信息表';
```

查看当前表为空。

```
select * from user_info;
```

2. **定义api请求接口/api/reg**



一般这种使用json封装请求的数据，具体见请求和应答的字段。

3. **使用apifox 工具发送http请求进行测试**

测试的时候也可以直接把请求发送的tc程序，具体见课上演示http://192.168.1.27:8081/api/reg。

请求和应答

注册是一个简单的HTTP接口，根据用户输入的注册信息，创建一个新的用户。

请求URL

URL	**http://192.168.1.27/api/reg**
请求方式	POST
HTTP版本	1.1
Content-Type	application/json

请求参数

参数名	**含义**	**规则说明**	**是否必须**	**缺省值**
email	邮箱	必须符合email规范	可选	无
firstPwd	密码	md5加密后的值	**必填**	无
nickName	用户昵称	不能超过32个字符	**必填**	无
phone	手机号码	不能超过16个字符	可选	无
userName	用户名称	不能超过32个字符	**必填**	无

返回结果参数说明

名称	含义	规则说明
code	结果值	0：成功；1：失败；2：用户存在

服务示例

调用接口

http://192.168.1.27/api/reg

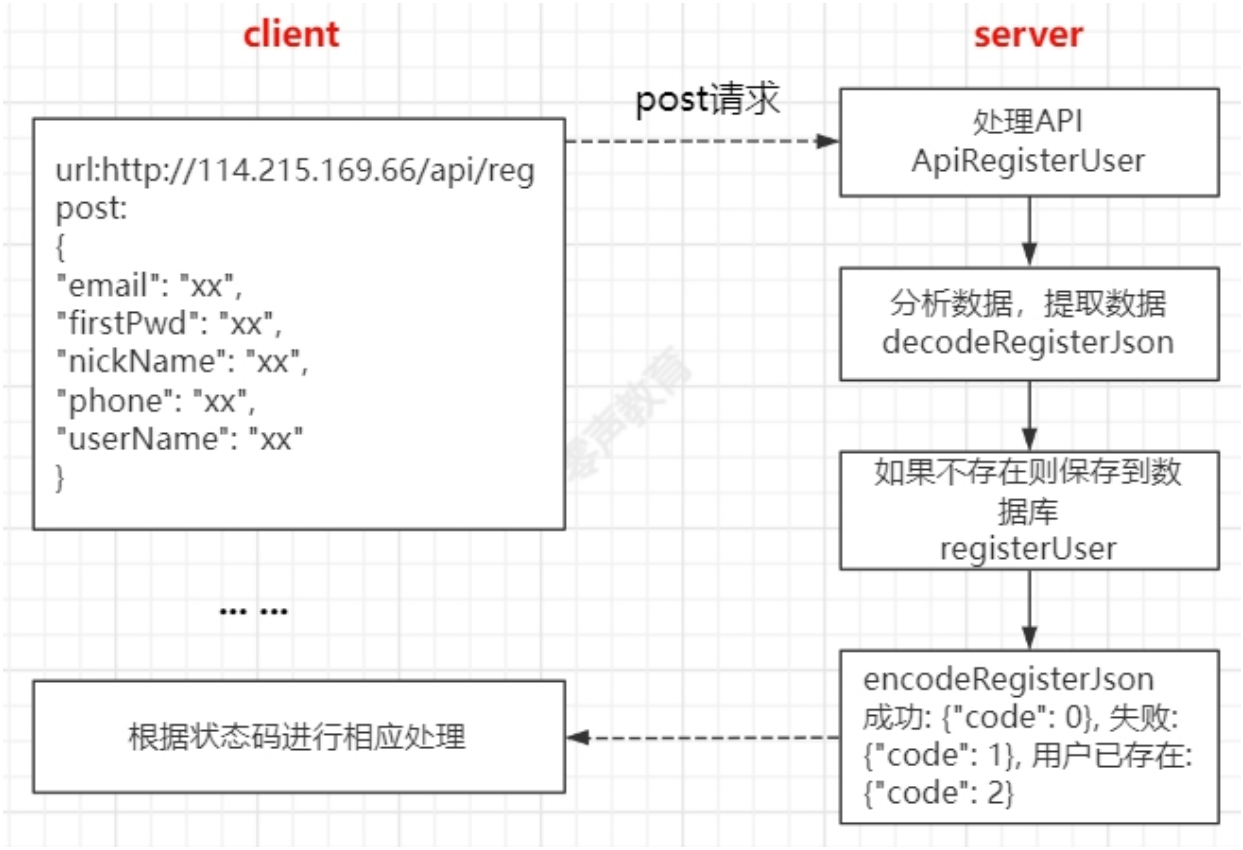
参数

```
{
"email": "472251823@qq.com",
"firstPwd": "e10adc3949ba59abbe56e057f20f883e",
"nickName": "lucky",
"phone": "18612345678",
"userName": "qingfu"
}
```

返回结果

```
{
"code": 0
}
```

处理逻辑



客户端提交的密码需要使用md5加密，不能提供明文的密码。

知识扩展

MD5

MD5即Message-Digest Algorithm 5（信息-摘要算法5），用于确保信息传输完整一致。是计算机广泛使用的杂凑算法之一（又译摘要算法、哈希算法），主流编程语言普遍已有MD5实现。

理论上MD5是不可逆的，而且MD5本来也不是作加密使用，而是用来校验数据的完整性，只是因为其不可逆且稳定、快速的特点，被广泛用于对明文密码的加密。

但是简单密码来说，破解者完全可以将一定范围内的密码字典全部计算出来之后存为数据库，之后直接查询进行破解。

用户重要信息(如密码)不应该明文保存到数据库，可以通过MD5加密后再保存：

id	user_name	nick_name	password
9	milo	milo	e10adc3949ba59abbe56e057f20f883e

盐值加密方式拓展

https://www.yuque.com/docs/share/9276a004-7bc0-4d15-a477-c88068fb37ba?# 《给密码加盐》

3.2 登录 /api/login api_login.cc

用户信息存储在user_info表，

查询user_info表

```
mysql> select * from user_info;
```

登录的时候需要用户名、密码和数据库的用户名、密码做对比。

如果对比识别返回错误，对比成功则把token也返回给客户端。

token用来后续请求做匹配校验（服务器的token存储在redis）。

请求和应答

登录，根据用户输入的登录信息，登录进入到后台系统。

请求URL

URL	http://192.168.1.27/api/login
请求方式	POST
HTTP版本	1.1
Content-Type	application/json

请求参数

参数名	含义	规则说明	是否必须	缺省值
pwd	密码	md5加密后的值	必填	无
user	用户名称	不能超过32个字符	必填	无

返回结果参数说明

名称	含义	规则说明
code	结果值	0： 成功1： 失败

名称	含义	规则说明
token	令牌	每次登录后，生成的token不一样，后续其他接口请求时，需要带上token。

服务示例

调用接口

http://192.168.1.27/api/login

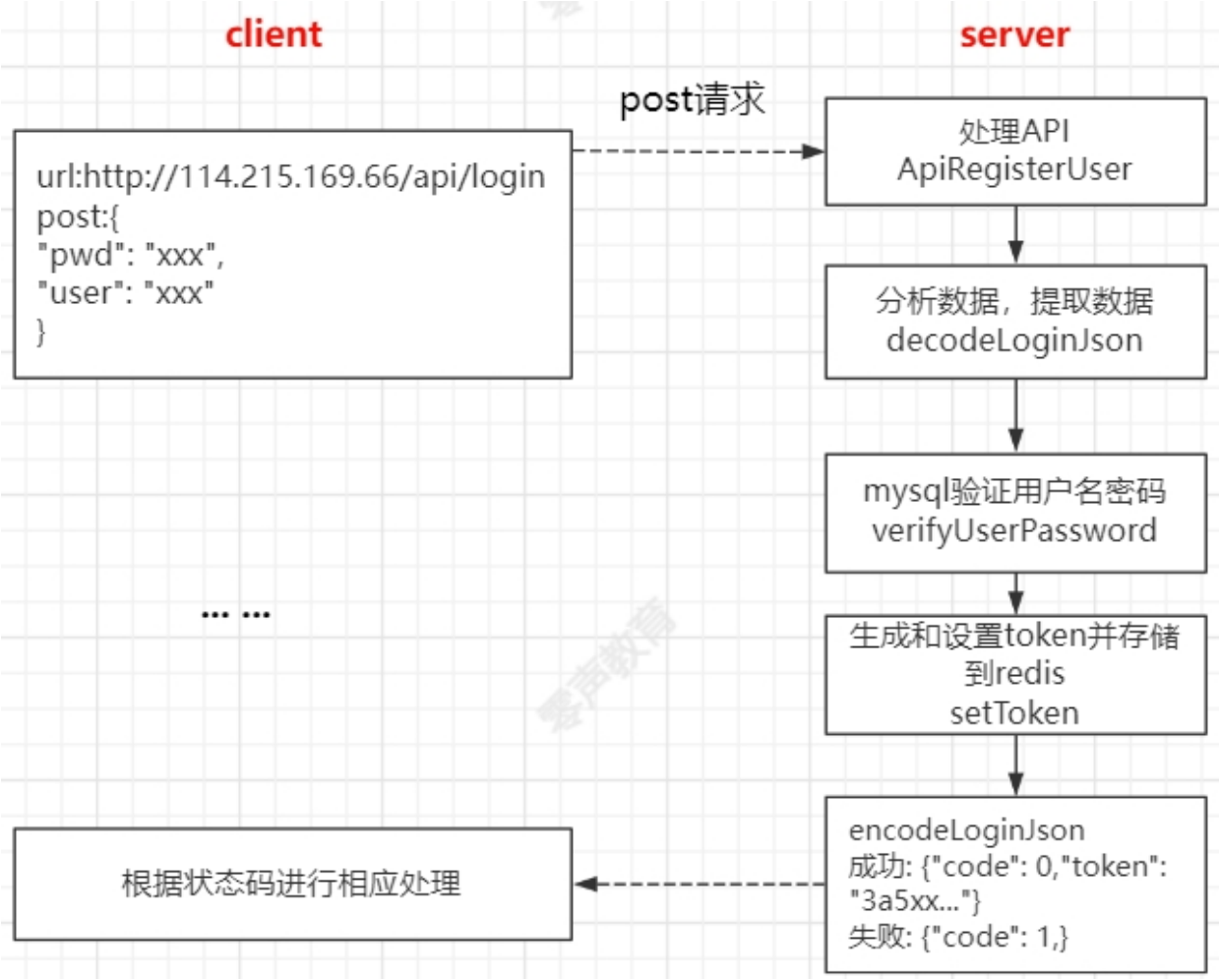
参数

```
{
"pwd": "e10adc3949ba59abbe56e057f20f883e",
"user": "qingfu"
}
```

返回结果

```
{
"code": 0,
"token": "3a58ca22317e637797f8bcad5c047446"
}
```

处理逻辑



知识扩展

Token验证

Token 的中文有人翻译成 “令牌”，意思就是，你拿着这个令牌，才能过一些关卡。

Token是一个用户自定义的任意字符串。在成功提交了开发者自定义的这个字符串之后，Token的值会保存到服务器后台。只有服务器和客户端前端知道这个字符串，于是Token就成了这两者之间的密钥，它可以让服务器确认请求是来自客户端还是恶意的第三方。

这里所说的Token，本质上就是http session：《http session介绍》

使用基于 Token 的身份验证方法，在服务端不需要存储用户的登录记录。大概的流程是这样的：

1. 客户端使用用户名跟密码请求登录
2. 服务端收到请求，去验证用户名与密码

3. 验证成功后，服务端生成一个 Token，这个Token可以存储在内存、磁盘、或者数据库里，再把这个 Token 发送给客户端
4. 客户端收到 Token 以后可以把它存储起来，比如放在 Cookie 里或者 Local Storage
5. 客户端每次向服务端请求资源的时候需要带着服务端签发的 Token
6. 服务端收到请求，然后去验证客户端请求里面带着的 Token，如果验证成功，就向客户端返回请求的数据

简单而言，本服务器程序是采用随机数+base64+md5生成的token，对于过期时间是通过redis的key超时机制实现。

更进一步的阅读：https://www.yuque.com/docs/share/18838587-4e59-4f2a-abd2-8a35471ead9d?#《还分不清 Cookie、Session、Token、JWT》

Base64

用记事本打开exe、jpg、pdf这些文件时，我们都会看到一大堆乱码，因为二进制文件包含很多无法显示和打印的字符。

当不可见字符在网络上传输时，比如说从 A 计算机传到 B 计算机，往往要经过多个路由设备，由于不同的设备对字符的处理方式有一些不同，这样那些不可见字符就有可能被处理错误，这是不利于传输的。

为了解决这个问题，我们可以先对数据进行编码，比如 编码，变成可见字符，也就是 ASCII 码可表示的可见字符，从而确保数据可靠传输。 的内容是有 0 ~ 9，a ~ z，A ~ Z，+，/ 组成，正好 64 个字符，这些字符是在 ASCII 可表示的范围内，属于 95 个可见字符的一部分。

所以，如果要让记事本这样的文本处理软件能处理二进制数据，如使用json保存二进制信息，需要先把数据先做一个Base64编码，统统变成可见字符，再保存。

在Base64中的可打印字符包括大写英文字母A-Z、小写英文字母a-z、阿拉伯数字0-9，这样共有62个字符，此外两个可打印符号在不同的系统中而不同，通常用加号（+）和正斜杠（/）。外加“补全符号”，通常用等号（=）。

Base64是一种用64个字符来表示任意二进制数据的方法，常用于在URL、Cookie、网页中传输少量二进制数据。

Base64要求把每三个8Bit的字节转换为四个6Bit的字节（3*8* = 4*6* = 24），然后把6Bit再添两位高位0，组成四个8Bit的字节，也就是说，转换后的字符串理论上将要比原来的长1/3。

(原文)转换前　　11111111, 11111111, 11111111　(二进制)

(Base64编码)转换后　　00111111, 00111111, 00111111, 00111111 （二进制）

3.3 用户文件列表/api/myfiles&cmd=normal api_myfiles.cc

开发逻辑

获取用户文件列表涉及到多表查询

1. 用户文件列表 (user_file_list)

字段名	中文含义	类型	是否允许空	默认值	其他
id	序号	int	否		自动递增，主键
user	文件所属用户	varchar(32)	否		
md5	文件md5	varchar(256)	否		
file_name	文件名字	varchar(128)	否	”	
shared_status	共享状态	int	否	0	0为没有共享， 1为共享
pv	文件下载量	int	否	0	默认值为0，下载一次加1
create_time	文件创建时间	timestamp	否	CURRENT_TIMESTAMP	

创建表命令：

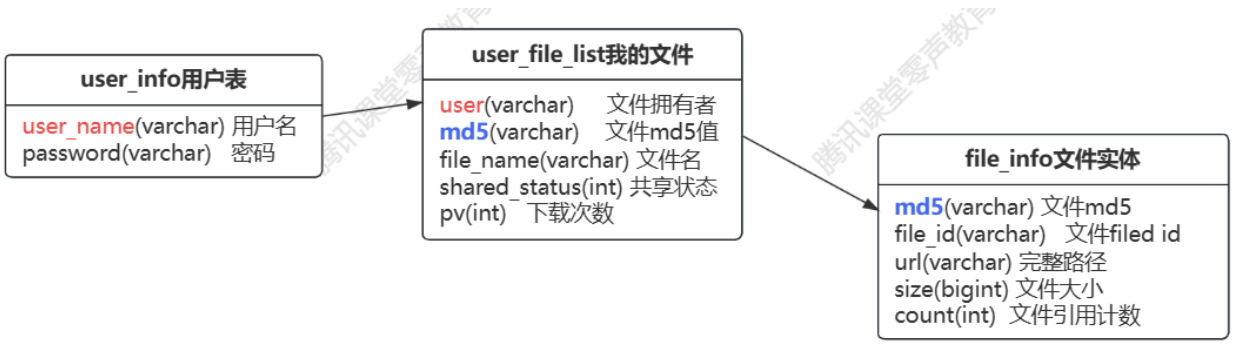
```sql
DROP TABLE IF EXISTS `user_file_list`;
CREATE TABLE `user_file_list` (
  `id` int(11) NOT NULL AUTO_INCREMENT COMMENT '编号',
  `user` varchar(32) NOT NULL COMMENT '文件所属用户',
  `md5` varchar(256) NOT NULL COMMENT '文件md5',
  `create_time` timestamp NULL DEFAULT CURRENT_TIMESTAMP COMMENT '文件创建时间',
  `file_name` varchar(128) DEFAULT NULL COMMENT '文件名字',
  `shared_status` int(11) DEFAULT NULL COMMENT '共享状态，0为没有共享， 1为共享',
  `pv` int(11) DEFAULT NULL COMMENT '文件下载量，默认值为0，下载一次加1',
  PRIMARY KEY (`id`),
  KEY   `idx_user_md5_file_name` (`user`,`md5`, `file_name`)
) ENGINE=InnoDB AUTO_INCREMENT=30 DEFAULT CHARSET=utf8 COMMENT='用户文件列表';
```

2. 文件信息表 (file_info)

字段名	中文含义	类型	是否允许空	默认值	其他
id	序号	bigint	否		自动递增，主键
md5	文件md5	varchar(256)	否	''	
file_id	文件id	varchar(256)	否	''	格式例如：/group1/M00/00/00/xxx.png
url	文件url	varchar(512)	否	''	格式例如：192.168.1.2:80/group1/M00/00/00/xxx.png
size	文件大小	bigint	否	0	以字节为单位
type	文件类型	varchar(32)	是	''	png, zip, mp4.....
count	文件引用计数	int	否	0	默认为1, 每增加一个用户拥有此文件，此计数器+1

创建表命令：

```sql
DROP TABLE IF EXISTS `file_info`;
CREATE TABLE `file_info` (
  `id` bigint(20) NOT NULL AUTO_INCREMENT COMMENT '文件序号，自动递增，主键',
  `md5` varchar(256) NOT NULL COMMENT '文件md5',
  `file_id` varchar(256) NOT NULL COMMENT '文件id:/group1/M00/00/00/xxx.png',
  `url` varchar(512) NOT NULL COMMENT '文件url
192.168.52.139:80/group1/M00/00/00/xxx.png',
  `size` bigint(20) DEFAULT '0' COMMENT '文件大小，以字节为单位',
  `type` varchar(32) DEFAULT '' COMMENT '文件类型： png，zip，mp4……',
  `count` int(11) DEFAULT '0' COMMENT '文件引用计数,默认为1。每增加一个用户拥有此文件，此计数
器+1',
  PRIMARY KEY (`id`),
  UNIQUE KEY `uq_md5` (`md5`)  COMMENT '前缀索引'
) ENGINE=InnoDB AUTO_INCREMENT=70 DEFAULT CHARSET=utf8 COMMENT='文件信息表';
```



生成测试数据，以qingfu用户名为例：

生成5条file_info文件信息

```sql
insert into file_info (md5, file_id, url, size, type, count) values
('6c5fa2864bb264c91167258b3e478fa0',
'group1/M00/00/00/eBuDxWCfQHSATopyAAV8AJV_1mw860.dll','http://192.168.1.27:80/group1
/M00/00/00/eBuDxWCfQHSATopyAAV8AJV_1mw860.dll', 1000, 'dll', 1);
insert into file_info (md5, file_id, url, size, type, count) values
('6c5fa2864bb264c91167258b3e478fa1',
'group1/M00/00/00/eBuDxWCfQHSATopyAAV8AJV_1mw861.dll','http://192.168.1.27:80/group1
/M00/00/00/eBuDxWCfQHSATopyAAV8AJV_1mw861.dll', 1000, 'dll', 1);
insert into file_info (md5, file_id, url, size, type, count) values
('6c5fa2864bb264c91167258b3e478fa2',
'group1/M00/00/00/eBuDxWCfQHSATopyAAV8AJV_1mw862.jpg','http://192.168.1.27:80/group1
/M00/00/00/eBuDxWCfQHSATopyAAV8AJV_1mw862.jpg', 1000, 'jpg', 1);
insert into file_info (md5, file_id, url, size, type, count) values
('6c5fa2864bb264c91167258b3e478fa3',
'group1/M00/00/00/eBuDxWCfQHSATopyAAV8AJV_1mw863.txt','http://192.168.1.27:80/group1
/M00/00/00/eBuDxWCfQHSATopyAAV8AJV_1mw863.txt', 1000, 'txt', 1);
insert into file_info (md5, file_id, url, size, type, count) values
('6c5fa2864bb264c91167258b3e478fa4',
'group1/M00/00/00/eBuDxWCfQHSATopyAAV8AJV_1mw864.mp4','http://192.168.1.27:80/group1
/M00/00/00/eBuDxWCfQHSATopyAAV8AJV_1mw864.mp4t', 1000, 'mp4', 1);
```

生成5条user_file_list文件信息

```
insert into user_file_list(user, md5, create_time, file_name, shared_status, pv)
values ('qingfu', '6c5fa2864bb264c91167258b3e478fa0', '2024-05-15 11:31:00',
'120.dll', 0, 0);
insert into user_file_list(user, md5, create_time, file_name, shared_status, pv)
values ('qingfu', '6c5fa2864bb264c91167258b3e478fa1', '2024-05-15 11:31:00',
'121.dll', 0, 0);
insert into user_file_list(user, md5, create_time, file_name, shared_status, pv)
values ('qingfu', '6c5fa2864bb264c91167258b3e478fa2', '2024-05-15 11:31:00',
'122.jpg', 0, 0);
insert into user_file_list(user, md5, create_time, file_name, shared_status, pv)
values ('qingfu', '6c5fa2864bb264c91167258b3e478fa3', '2024-05-15 11:31:00',
'123.txt', 0, 0);
insert into user_file_list(user, md5, create_time, file_name, shared_status, pv)
values ('qingfu', '6c5fa2864bb264c91167258b3e478fa4', '2024-05-15 11:31:00',
'124.mp4', 0, 0);
```

获取文件数量

```
select count(*) from user_file_list where user='qingfu';
```

请求和应答

请求URL

URL	http://192.168.1.27/api/myfiles?cmd=normal
请求方式	POST
HTTP版本	1.1
Content-Type	application/json

请求参数

参数名	含义	规则说明	是否必须	缺省值
token	令牌	同上	必填	无
user	用户名称	不能超过32个字符	必填	无
count	文件个数	文件个数需大于0	必填	无
start	开始位置		必填	无

返回结果参数说明

名称	含义	规则说明
files	文件结果集	"code": 0正常，1失败"count": 返回的当前文件数量，比如2"total": 个人文件总共的数量"user": 用户名称,"md5": md5值,"create_time": 创建时间,"file_name": 文件名,"share_status": 共享状态, 0为没有共享， 1为共享"pv": 文件下载量，下载一次加1"url": URL,"size": 文件大小,"type": 文件类型

服务示例

调用接口

http://192.168.1.27/api/myfiles?cmd=normal

参数

```
{
"count": 10,
"start": 0,
"token": "nwlrbbmqbhcdarzowkkyhiddqscdxrjm",
"user": "qingfu"
}
```

返回结果，这里只是示例，具体以实际返回结果为准。

```
{
    "code": 0,
    "count": 5,
    "files": [
        {
            "create_time": "2024-05-15 11:31:00",
            "file_name": "120.dll",
            "md5": "6c5fa2864bb264c91167258b3e478fa0",
            "pv": 0,
            "share_status": 0,
            "size": 1000,
            "type": "dll",
            "url":
"http://192.168.1.27:80/group1/M00/00/00/eBuDxWCfQHSATopyAAV8AJV_1mw860.dll",
            "user": "qingfu"
        },
        {
            "create_time": "2024-05-15 11:31:00",
            "file_name": "121.dll",
            "md5": "6c5fa2864bb264c91167258b3e478fa1",
            "pv": 0,
            "share_status": 0,
            "size": 1000,
```

```
            "type": "dll",
            "url":
"http://192.168.1.27:80/group1/M00/00/00/eBuDxWCfQHSATopyAAV8AJV_1mw861.dll",
            "user": "qingfu"
        },
        {
            "create_time": "2024-05-15 11:31:00",
            "file_name": "122.jpg",
            "md5": "6c5fa2864bb264c91167258b3e478fa2",
            "pv": 0,
            "share_status": 0,
            "size": 1000,
            "type": "jpg",
            "url":
"http://192.168.1.27:80/group1/M00/00/00/eBuDxWCfQHSATopyAAV8AJV_1mw862.jpg",
            "user": "qingfu"
        },
        {
            "create_time": "2024-05-15 11:31:00",
            "file_name": "123.txt",
            "md5": "6c5fa2864bb264c91167258b3e478fa3",
            "pv": 0,
            "share_status": 0,
            "size": 1000,
            "type": "txt",
            "url":
"http://192.168.1.27:80/group1/M00/00/00/eBuDxWCfQHSATopyAAV8AJV_1mw863.txt",
            "user": "qingfu"
        },
        {
            "create_time": "2024-05-15 11:31:00",
            "file_name": "124.mp4",
            "md5": "6c5fa2864bb264c91167258b3e478fa4",
            "pv": 0,
            "share_status": 0,
            "size": 1000,
            "type": "mp4",
            "url":
"http://192.168.1.27:80/group1/M00/00/00/eBuDxWCfQHSATopyAAV8AJV_1mw864.mp4t",
            "user": "qingfu"
        }
    ],
    "total": 5
}
```


文件信息字段：

- code: 0正常，1错误
- user：文件上传者

- md5： 文件md5码
- time：文件上传的时间
- filename： 用户名
- share_status： 文件是否已共享，已经共享值为1，否则值为0
- pv： 文件当前的下载量，下载一次，pv++
- url： 文件存放在fastdfs分布式存储服务器的url地址
- size： 文件大小
- type： 文件类型

处理逻辑