Tedu.cn
达内教育

《分布式流媒体》实训项目

C/C++教学体系

Tedu.cn
达内教育

TNV DAY12

直播课

目录

文件操作类(file_c)

属性、构造和析构

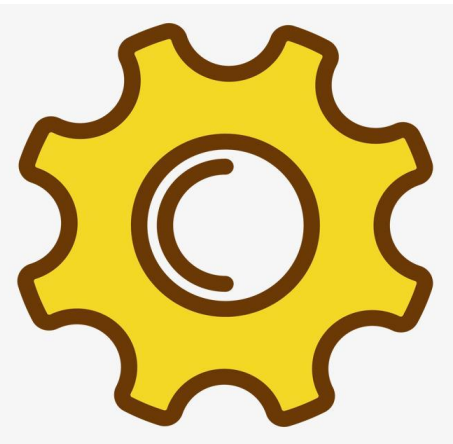
知识讲解

- 成员变量
  - 文件描述符：m_fd
  - 打开标志
    - 读：O_READ
    - 写：O_WRITE
- 构造函数
  - 初始文件描述符为-1
- 析构函数
  - 关闭文件

方法

知识讲解

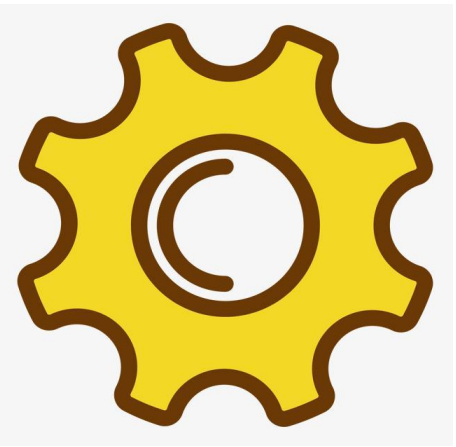
- 打开文件：open
  - 检查路径
  - 打开文件
  - 检查失败
- 关闭文件：close
  - 关闭文件
  - 复位文件描述符为-1
- 读取文件：read
  - 检查文件描述符
  - 检查文件缓冲区
  - 读取给定字节数



方法

知识讲解

- 写入文件：write
  - 检查文件描述符
  - 检查文件缓冲区
  - 写入给定字节数
- 设置偏移：seek
  - 检查文件描述符
  - 检查文件偏移量
  - 设置文件偏移量
- 删除文件(静态)：del
  - 检查路径
  - 删除文件

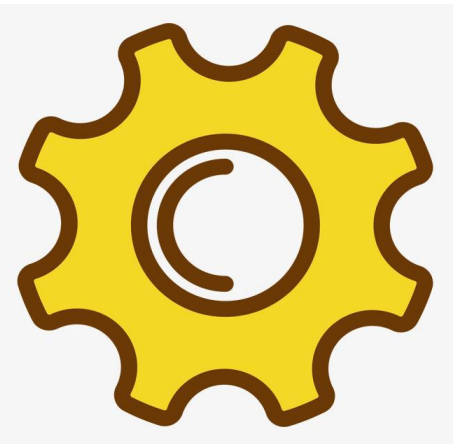


ID客户机类(id_c)

方法

知识讲解

- 从ID服务器获取与ID键相对应的值：get
  - 检查ID键
  - 构造请求
  - 向ID服务器发送请求，接收并解析响应，从中获得ID值
- 向ID服务器发送请求，接收并解析响应，从中获得ID值：client
  - 从ID服务器地址表中随机抽取一台ID服务器尝试连接
  - 向ID服务器发送请求
  - 从ID服务器接收响应
  - 从ID服务器的响应中解析出ID值



业务服务类(service_c)

一级方法

知识讲解

- 业务处理：business
  - 解析包头
    - 包体长度
    - 命令
    - 状态
  - 根据命令执行具体业务处理
    - 处理来自客户机的上传文件请求
    - 处理来自客户机的询问文件大小请求
    - 处理来自客户机的下载文件请求
    - 处理来自客户机的删除文件请求
  - 返回处理结果

二级方法

- 处理来自客户机的上传文件请求：upload
  - 检查包体长度
  - 接收包体
  - 解析包体
  - 检查文件大小
  - 生成文件路径
  - 接收并保存文件
  - 响应成功报文

包头			包体				
包体长度	命令(70)	状态	应用ID	用户ID	文件ID	文件大小	文件内容
8	1	1	16	256	128	8	文件大小

知识讲解

二级方法

知识讲解

- 处理来自客户机的询问文件大小请求：filesize
  - 检查包体长度
  - 接收包体
  - 解析包体
  - 实例化数据库访问对象
  - 连接数据库
  - 根据文件ID获取其对应的路径及大小
  - 构造响应
  - 发送响应
  - 返回成功

包头			包体		
包体长度	命令(71)	状态	应用ID	用户ID	文件ID
8	1	1	16	256	128

包头			包体
包体长度	命令(102)	状态	文件大小
8	1	1	8

二级方法

知识讲解

- 处理来自客户机的下载文件请求：download
  - 检查包体长度
  - 接收包体
  - 解析包体
  - 实例化数据库访问对象
  - 连接数据库
  - 根据文件ID获取其对应的路径及大小
  - 检查位置
  - 大小为零表示下到文件尾，检查大小
  - 读取并发送文件
  - 返回成功

包头			包体				
包体长度	命令(72)	状态	应用ID	用户ID	文件ID	偏移	大小
8	1	1	16	256	128	8	8

包头			包体
包体长度	命令(102)	状态	文件内容
8	1	1	大小

二级方法

知识讲解

- 处理来自客户机的删除文件请求：del
  - 检查包体长度
  - 接收包体
  - 解析包体
  - 实例化数据库访问对象
  - 连接数据库
  - 根据文件ID获取其对应的路径及大小
  - 删除文件ID
  - 删除文件
  - 响应成功报文

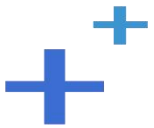
包头			包体		
包体长度	命令(73)	状态	应用ID	用户ID	文件ID
8	1	1	16	256	128

附录：程序清单

TNV/src/04_storage/07_file.h

知识讲解

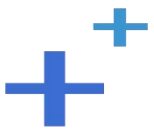
```
//存储服务器
//声明文件操作类
//
#pragma once

#include <sys/types.h>
//
//文件操作类
//
class file_c {
public:
        //构造函数
        file_c(void);
        //析构函数
        ~file_c(void);
```



TNV/src/04_storage/07_file.h

知识讲解

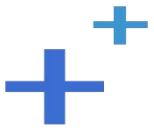
```
//打开文件
int open(char const* path, char flag);
//关闭文件
int close(void);

//读取文件
int read(void* buf, size_t count) const;
//写入文件
int write(void const* buf, size_t count) const;

//设置偏移
```



TNV/src/04_storage/07_file.h

知识讲解

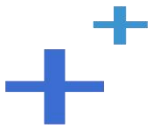
```
        int seek(off_t offset) const;
        //删除文件
        static int del(char const* path);

        //打开标志
        static char const O_READ   = 'r';
        static char const O_WRITE  = 'w';

private:
        int m_fd; //文件描述符
};
```



TNV/src/04_storage/08_file.cpp

知识讲解

```
//存储服务器
//实现文件操作类
//
#include <unistd.h>
#include <fcntl.h>
#include <lib_acl.hpp>
#include "01_types.h"
#include "07_file.h"

//构造函数
file_c::file_c(void): m_fd(-1) {
}

//析构函数
file_c::~file_c(void) {
```



TNV/src/04_storage/08_file.cpp

知识讲解

```
        close();
}

//打开文件
int file_c::open(char const* path, char flag) {
        //检查路径
        if (!path || !strlen(path)) {
                logger_error("path is null");
                return ERROR;
        }

        //打开文件
        if (flag == O_READ)
                m_fd = ::open(path, O_RDONLY);
        else if (flag == O_WRITE)
```

TNV/src/04_storage/08_file.cpp

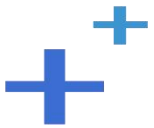
```
            m_fd = ::open(path, O_WRONLY | O_CREAT | O_TRUNC, 0644);
        else {
            logger_error("unknown open flag: %c", flag);
            return ERROR;
        }

        //打开失败
        if (m_fd == -1) {
            logger_error("call open fail: %s, path: %s, flag: %c",
                    strerror(errno), path, flag);
            return ERROR;
        }

        return OK;
    }
```

知识讲解

TNV/src/04_storage/08_file.cpp

知识讲解

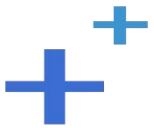
```
//关闭文件
int file_c::close(void) {
        if (m_fd != -1) {
                //关闭文件
                if (::close(m_fd) == -1) {
                        logger_error("call close fail: %s", strerror(errno));
                        return ERROR;
                }

                m_fd = -1;
        }

        return OK;
}
```



TNV/src/04_storage/08_file.cpp

知识讲解

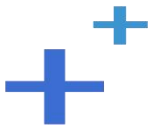
```
//读取文件
int file_c::read(void* buf, size_t count) const {
        //检查文件描述符
        if (m_fd == -1) {
                logger_error("invalid file handle");
                return ERROR;
        }

        //检查文件缓冲区
        if (!buf) {
                logger_error("invalid file buffer");
                return ERROR;
        }

        //读取给定字节数
```



TNV/src/04_storage/08_file.cpp

知识讲解

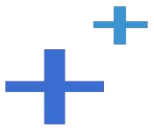
```
        ssize_t bytes = ::read(m_fd, buf, count);
        if (bytes == -1) {
                logger_error("call read fail: %s", strerror(errno));
                return ERROR;
        }
        if ((size_t)bytes != count) {
                logger_error("unable to read expected bytes: %ld != %lu",
                        bytes, count);
                return ERROR;
        }

        return OK;
}

//写入文件
```



TNV/src/04_storage/08_file.cpp

知识讲解

```
int file_c::write(void const* buf, size_t count) const {
        //检查文件描述符
        if (m_fd == -1) {
                logger_error("invalid file handle");
                return ERROR;
        }

        //检查文件缓冲区
        if (!buf) {
                logger_error("invalid file buffer");
                return ERROR;
        }

        //写入给定字节数
        if (::write(m_fd, buf, count) == -1) {
```



TNV/src/04_storage/08_file.cpp

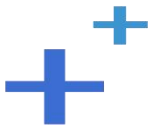
知识讲解

```
                logger_error("call write fail: %s", strerror(errno));
                return ERROR;
        }

        return OK;
}

//设置偏移
int file_c::seek(off_t offset) const {
        //检查文件描述符
        if (m_fd == -1) {
                logger_error("invalid file handle");
                return ERROR;
        }
```

TNV/src/04_storage/08_file.cpp

知识讲解

```
    //检查文件偏移量
    if (offset < 0) {
            logger_error("invalid file offset");
            return ERROR;
    }

    //设置文件偏移量
    if (lseek(m_fd, offset, SEEK_SET) == -1) {
            logger_error("call lseek fail: %s, offset: %ld",
                    strerror(errno), offset);
            return ERROR;
    }

    return OK;
}
```



TNV/src/04_storage/08_file.cpp

知识讲解

```
//删除文件
int file_c::del(char const* path) {
        //检查路径
        if (!path || !strlen(path)) {
                logger_error("path is null");
                return ERROR;
        }

        //删除文件
        if (unlink(path) == -1) {
                logger_error("call unlink fail: %s, path: %s",
                        strerror(errno), path);
                return ERROR;
        }

        return OK;
}
```

TNV/src/04_storage/09_id.h

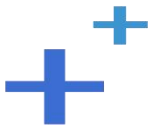
知识讲解

```
// 存储服务器
// 声明ID客户机类
//
#pragma once
//
// ID客户机类
//
class id_c {
public:
        // 从ID服务器获取与ID键相对应的值
        long get(char const* key) const;

private:
        // 向ID服务器发送请求，接收并解析响应，从中获得ID值
        long client(char const* requ, long long requlen) const;
};
```

TNV/src/04_storage/10_id.cpp

知识讲解

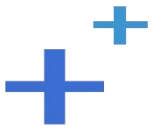
```
// 存储服务器
// 实现ID客户机类
//
#include <lib_acl.hpp>
#include "02_proto.h"
#include "03_util.h"
#include "01_globals.h"
#include "09_id.h"

// 从ID服务器获取与ID键相对应的值
long id_c::get(char const* key) const {
        // 检查ID键
        if (!key) {
                logger_error("key is null");
                return -1;
```



TNV/src/04_storage/10_id.cpp

知识讲解

```
}
size_t keylen = strlen(key);
if (!keylen) {
        logger_error("key is null");
        return -1;
}
if (keylen > ID_KEY_MAX) {
        logger_error("key too big: %lu > %d", keylen, ID_KEY_MAX);
        return -1;
}

//|包体长度|命令|状态| ID键 |
//|  8  | 1| 1|包体长度|
//构造请求
long long bodylen = keylen + 1;
```



TNV/src/04_storage/10_id.cpp

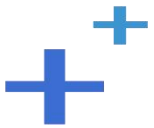
知识讲解

```
        long long requlen = HEADLEN + bodylen;
        char requ[requlen] = {};
        llton(bodylen, requ);
        requ[BODYLEN_SIZE] = CMD_ID_GET;
        requ[BODYLEN_SIZE+COMMAND_SIZE] = 0;
        strcpy(requ + HEADLEN, key);

        // 向ID服务器发送请求，接收并解析响应，从中获得ID值
        return client(requ, requlen);
}

// 向ID服务器发送请求，接收并解析响应，从中获得ID值
long id_c::client(char const* requ, long long requlen) const {
        acl::socket_stream conn;
```

TNV/src/04_storage/10_id.cpp

知识讲解

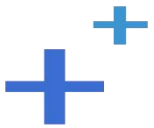
```
// 从ID服务器地址表中随机抽取一台ID服务器尝试连接
srand(time(NULL));
int nids = g_iaddrs.size();
int nrand = rand() % nids;
for (int i = 0; i < nids; ++i)
        if (conn.open(g_iaddrs[nrand].c_str(), 0, 0)) {
                logger("connect id success, addr: %s",
                        g_iaddrs[nrand].c_str());
                break;
        }
        else {
                logger("connect id fail, addr: %s",
                        g_iaddrs[nrand].c_str());
                nrand = (nrand + 1) % nids;
        }
```



TNV/src/04_storage/10_id.cpp

知识讲解

```
if (!conn.alive()) {
        logger_error("connect id fail, addrs: %s", cfg_iaddrs);
        return -1;
}

// 向ID服务器发送请求
if (conn.write(requ, requlen) < 0) {
        logger_error("write fail: %s, requlen: %lld, to: %s",
                acl::last_serror(), requlen, conn.get_peer());
        conn.close();
        return -1;
}

// 从ID服务器接收响应
long long resplen = HEADLEN + BODYLEN_SIZE;
```



TNV/src/04_storage/10_id.cpp

知识讲解

```
char resp[resplen] = {};
if (conn.read(resp, resplen) < 0) {
        logger_error("read fail: %s, resplen: %lld, from: %s",
                acl::last_serror(), resplen, conn.get_peer());
        conn.close();
        return -1;
}

// |包体长度|命令|状态|ID值|
// |    8   |  1 |  1 |  8 |
// 从ID服务器的响应中解析出ID值
long value = ntol(resp + HEADLEN);

conn.close();

return value;
}
```

TNV/src/04_storage/11_service.h

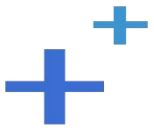
知识讲解

```
//存储服务器
//声明业务服务类
//
#pragma once

#include <lib_acl.hpp>
//
//业务服务类
//
class service_c {
public:
        //业务处理
        bool business(acl::socket_stream* conn, char const* head) const;

private:
```

TNV/src/04_storage/11_service.h

知识讲解

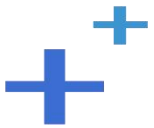
```
//处理来自客户机的上传文件请求
bool upload(acl::socket_stream* conn, long long bodylen) const;
//处理来自客户机的询问文件大小请求
bool filesize(acl::socket_stream* conn, long long bodylen) const;
//处理来自客户机的下载文件请求
bool download(acl::socket_stream* conn, long long bodylen) const;
//处理来自客户机的删除文件请求
bool del(acl::socket_stream* conn, long long bodylen) const;

//生成文件路径
int genpath(char* filepath) const;
//将ID转换为512进制
long id512(long id) const;
//用文件ID生成文件路径
int id2path(char const* spath, long fileid, char* filepath) const;
```



TNV/src/04_storage/11_service.h

知识讲解

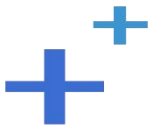
```
        //接收并保存文件
        int save(acl::socket_stream* conn, char const* appid,
                char const* userid, char const* fileid, long long filesize,
                char const* filepath) const;

        //读取并发送文件
        int send(acl::socket_stream* conn, char const* filepath,
                long long offset, long long size) const;

        //应答成功
        bool ok(acl::socket_stream* conn) const;
        //应答错误
        bool error(acl::socket_stream* conn, short errnumb,
                char const* format, ...) const;
};
```



TNV/src/04_storage/12_service.cpp

知识讲解

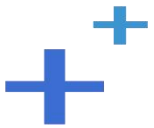
```
//存储服务器
//实现业务服务类
//
#include <linux/limits.h>
#include <algorithm>
#include "02_proto.h"
#include "03_util.h"
#include "01_globals.h"
#include "05_db.h"
#include "07_file.h"
#include "09_id.h"
#include "11_service.h"

//业务处理
bool service_c::business(acl::socket_stream* conn,
```



TNV/src/04_storage/12_service.cpp

知识讲解

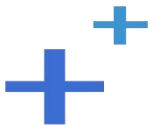
```
char const* head) const {
//|包体长度|命令|状态| 包体 |
//|  8  | 1| 1|包体长度|
//解析包头
long long bodylen = ntoll(head); //包体长度
if (bodylen < 0) {
        error(conn, -1, "invalid body length: %lld < 0", bodylen);
        return false;
}
int command = head[BODYLEN_SIZE]; //命令
int status = head[BODYLEN_SIZE+COMMAND_SIZE]; //状态
logger("bodylen: %lld, command: %d, status: %d",
        bodylen, command, status);

bool result;
```



TNV/src/04_storage/12_service.cpp

知识讲解

```
//根据命令执行具体业务处理
switch (command) {
        case CMD_STORAGE_UPLOAD:
                //处理来自客户机的上传文件请求
                result = upload(conn, bodylen);
                break;

        case CMD_STORAGE_FILESIZE:
                //处理来自客户机的询问文件大小请求
                result = filesize(conn, bodylen);
                break;

        case CMD_STORAGE_DOWNLOAD:
                //处理来自客户机的下载文件请求
                result = download(conn, bodylen);
```



TNV/src/04_storage/12_service.cpp

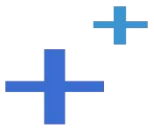
知识讲解

```
                break;

            case CMD_STORAGE_DELETE:
                //处理来自客户机的删除文件请求
                result = del(conn, bodylen);
                break;

            default:
                error(conn, -1, "unknown command: %d", command);
                return false;
        }

        return result;
    }
```

TNV/src/04_storage/12_service.cpp

知识讲解

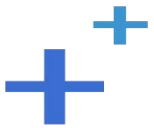
```
////////////////////////////////////////////////////////////////////////

//处理来自客户机的上传文件请求
bool service_c::upload(acl::socket_stream* conn,
        long long bodylen) const {
        //|包体长度|命令|状态|应用ID|用户ID|文件ID|文件大小|文件内容|
        //|  8  | 1| 1| 16 | 256| 128|  8  |文件大小|
        //检查包体长度
        long long expected = APPID_SIZE + USERID_SIZE +FILEID_SIZE +
                BODYLEN_SIZE;
        if (bodylen < expected) {
                error(conn, -1, "invalid body length: %lld < %lld",
                        bodylen, expected);
                return false;
        }
```



TNV/src/04_storage/12_service.cpp

知识讲解

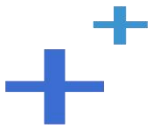
```
//接收包体
char body[expected];
if (conn->read(body, expected) < 0) {
        logger_error("read fail: %s, expected: %lld, from: %s",
                acl::last_serror(), expected, conn->get_peer());
        return false;
}

//解析包体
char appid[APPID_SIZE];
strcpy(appid, body);
char userid[USERID_SIZE];
strcpy(userid, body + APPID_SIZE);
char fileid[FILEID_SIZE];
strcpy(fileid, body + APPID_SIZE + USERID_SIZE);
```



TNV/src/04_storage/12_service.cpp

知识讲解

```
long long filesize = ntoll(
        body + APPID_SIZE + USERID_SIZE + FILEID_SIZE);

//检查文件大小
if (filesize != bodylen - expected) {
        logger_error("invalid file size: %lld != %lld",
                filesize, bodylen - expected);
        error(conn, -1, "invalid file size: %lld != %lld",
                filesize, bodylen - expected);
        return false;
}

//生成文件路径
char filepath[PATH_MAX+1];
if (genpath(filepath) != OK) {
```



TNV/src/04_storage/12_service.cpp

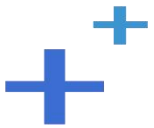
知识讲解

```
            error(conn, -1, "get filepath fail");
            return false;
    }

    logger("upload file, appid: %s, userid: %s, "
            "fileid: %s, filesize: %lld, filepath: %s",
            appid, userid, fileid, filesize, filepath);

    //接收并保存文件
    int result = save(conn, appid, userid, fileid, filesize, filepath);
    if (result == SOCKET_ERROR)
            return false;
    else if (result == ERROR) {
            error(conn, -1, "receive and save file fail, fileid: %s",
                    fileid);
```

TNV/src/04_storage/12_service.cpp

知识讲解

```
                return false;
        }

        return ok(conn);
}

//处理来自客户机的询问文件大小请求
bool service_c::filesize(acl::socket_stream* conn,
        long long bodylen) const {
        //|包体长度|命令|状态|应用ID|用户ID|文件ID|
        //|    8    |  1 |  1 |  16  |  256 |  128 |
        //检查包体长度
        long long expected = APPID_SIZE + USERID_SIZE + FILEID_SIZE;
        if (bodylen != expected) {
                error(conn, -1, "invalid body length: %lld != %lld",
```



TNV/src/04_storage/12_service.cpp

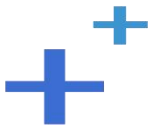
知识讲解

```
                bodylen, expected);
        return false;
}

//接收包体
char body[bodylen];
if (conn->read(body, bodylen) < 0) {
        logger_error("read fail: %s, bodylen: %lld, from: %s",
                acl::last_serror(), bodylen, conn->get_peer());
        return false;
}

//解析包体
char appid[APPID_SIZE];
strcpy(appid, body);
```

TNV/src/04_storage/12_service.cpp

知识讲解

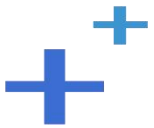
```
char userid[USERID_SIZE];
strcpy(userid, body + APPID_SIZE);
char fileid[FILEID_SIZE];
strcpy(fileid, body + APPID_SIZE + USERID_SIZE);

db_c db; //数据库访问对象

//连接数据库
if (db.connect() != OK)
        return false;

std::string filepath; //文件路径
long long   filesize; //文件大小

//根据文件ID获取其对应的路径及大小
```



TNV/src/04_storage/12_service.cpp

知识讲解

```
if (db.get(appid, userid, fileid, filepath, &filesize) != OK) {
        error(conn, -1, "read database fail, fileid: %s", fileid);
        return false;
}

logger("filesize, appid: %s, userid: %s, "
        "fileid: %s, filepath: %s, filesize: %lld",
        appid, userid, fileid, filepath.c_str(), filesize);

//|包体长度|命令|状态|文件大小|
//|    8   |  1 |  1 |    8   |
//构造响应
bodylen = BODYLEN_SIZE;
long long resplen = HEADLEN + bodylen;
char resp[resplen] = {};
```



TNV/src/04_storage/12_service.cpp

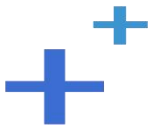
知识讲解

```
        llton(bodylen, resp);
        resp[BODYLEN_SIZE] = CMD_STORAGE_REPLY;
        resp[BODYLEN_SIZE+COMMAND_SIZE] = 0;
        llton(filesize, resp + HEADLEN);

        //发送响应
        if (conn->write(resp, resplen) < 0) {
                logger_error("write fail: %s, resplen: %lld, to: %s",
                        acl::last_serror(), resplen, conn->get_peer());
                return false;
        }

        return true;
}
```

TNV/src/04_storage/12_service.cpp

知识讲解

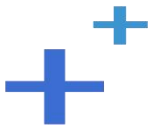
```
//处理来自客户机的下载文件请求
bool service_c::download(acl::socket_stream* conn,
        long long bodylen) const {
        //|包体长度|命令|状态|应用ID|用户ID|文件ID|偏移|大小|
        //|    8   | 1 | 1 |  16  |  256 |  128 |  8 |  8 |
        //检查包体长度
        long long expected = APPID_SIZE + USERID_SIZE + FILEID_SIZE +
                BODYLEN_SIZE + BODYLEN_SIZE;
        if (bodylen != expected) {
                error(conn, -1, "invalid body length: %lld != %lld",
                        bodylen, expected);
                return false;
        }

        //接收包体
```



TNV/src/04_storage/12_service.cpp

知识讲解

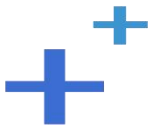
```
char body[bodylen];
if (conn->read(body, bodylen) < 0) {
        logger_error("read fail: %s, bodylen: %lld, from: %s",
                acl::last_serror(), bodylen, conn->get_peer());
        return false;
}

//解析包体
char appid[APPID_SIZE];
strcpy(appid, body);
char userid[USERID_SIZE];
strcpy(userid, body + APPID_SIZE);
char fileid[FILEID_SIZE];
strcpy(fileid, body + APPID_SIZE + USERID_SIZE);
long long offset = ntoll(
```



TNV/src/04_storage/12_service.cpp

知识讲解

```
        body + APPID_SIZE + USERID_SIZE + FILEID_SIZE);
long long size = ntoll(
        body + APPID_SIZE + USERID_SIZE + FILEID_SIZE + BODYLEN_SIZE);

db_c db; //数据库访问对象

//连接数据库
if (db.connect() != OK)
        return false;

std::string filepath; //文件路径
long long   filesize; //文件大小

//根据文件ID获取其对应的路径及大小
if (db.get(appid, userid, fileid, filepath, &filesize) != OK) {
```



TNV/src/04_storage/12_service.cpp

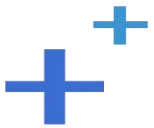
知识讲解

```
            error(conn, -1, "read database fail, fileid: %s", fileid);
            return false;
    }

    //检查位置
    if (offset < 0 || filesize < offset) {
            logger_error("invalid offset, %lld is not between 0 and %lld",
                    offset, filesize);
            error(conn, -1, "invalid offset, %lld is not between 0 and %lld",
                    offset, filesize);
            return false;
    }

    //大小为零表示下到文件尾
    if (!size)
```

TNV/src/04_storage/12_service.cpp

知识讲解

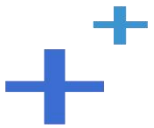
```
        size = filesize - offset;

    //检查大小
    if (size < 0 || filesize - offset < size) {
        logger_error("invalid size, %lld is not between 0 and %lld",
                size, filesize - offset);
        error(conn, -1, "invalid offset, %lld is not between 0 and %lld",
                size, filesize - offset);
        return false;
    }

    logger("download file, appid: %s, userid: %s, fileid: %s, "
        "offset: %lld, size: %lld, filepath: %s, filesize: %lld",
        appid, userid, fileid, offset, size, filepath.c_str(), filesize);
```



TNV/src/04_storage/12_service.cpp

知识讲解

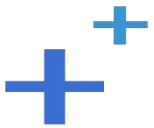
```
        //读取并发送文件
        int result = send(conn, filepath.c_str(), offset, size);
        if (result == SOCKET_ERROR)
                return false;
        else if (result == ERROR) {
                error(conn, -1, "read and send file fail, fileid: %s", fileid);
                return false;
        }

        return true;
}

//处理来自客户机的删除文件请求
bool service_c::del(acl::socket_stream* conn, long long bodylen) const {
        //|包体长度|命令|状态|应用ID|用户ID|文件ID|
```



TNV/src/04_storage/12_service.cpp

知识讲解

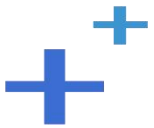
```
// |    8    |  1  |  1  |   16   |   256  |   128  |
//检查包体长度
long long expected = APPID_SIZE + USERID_SIZE + FILEID_SIZE;
if (bodylen != expected) {
        error(conn, -1, "invalid body length: %lld != %lld",
                bodylen, expected);
        return false;
}

//接收包体
char body[bodylen];
if (conn->read(body, bodylen) < 0) {
        logger_error("read fail: %s, bodylen: %lld, from: %s",
                acl::last_serror(), bodylen, conn->get_peer());
        return false;
```



TNV/src/04_storage/12_service.cpp

知识讲解

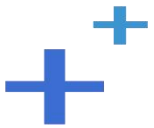
```
}

//解析包体
char appid[APPID_SIZE];
strcpy(appid, body);
char userid[USERID_SIZE];
strcpy(userid, body + APPID_SIZE);
char fileid[FILEID_SIZE];
strcpy(fileid, body + APPID_SIZE + USERID_SIZE);

db_c db; //数据库访问对象

//连接数据库
if (db.connect() != OK)
        return false;
```



TNV/src/04_storage/12_service.cpp

知识讲解

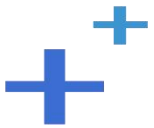
```
std::string filepath; //文件路径
long long   filesize; //文件大小

//根据文件ID获取其对应的路径及大小
if (db.get(appid, userid, fileid, filepath, &filesize) != OK) {
        error(conn, -1, "read database fail, fileid: %s", fileid);
        return false;
}

//删除文件ID
if (db.del(appid, userid, fileid) != OK) {
        error(conn, -1, "delete database fail, fileid: %s", fileid);
        return false;
}
```



TNV/src/04_storage/12_service.cpp

知识讲解

```
        //删除文件
        if (file_c::del(filepath.c_str()) != OK) {
                error(conn, -1, "delete file fail, fileid: %s", fileid);
                return false;
        }

        logger("delete file success, appid: %s, userid: %s, "
                "fileid: %s, filepath: %s, filesize: %lld",
                appid, userid, fileid, filepath.c_str(), filesize);

        return ok(conn);
}

////////////////////////////////////////////////////////////////////////
```



复习课见