Tedu.cn
达内教育

《分布式流媒体》实训项目

C/C++教学体系

TNV DAY09

直播课

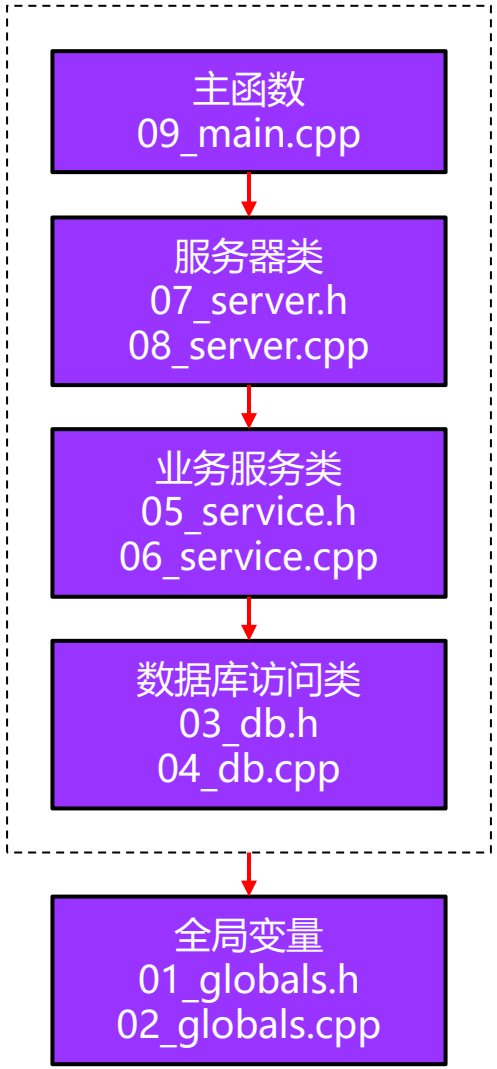
目录

ID服务器详细设计

组织结构

- 03_id

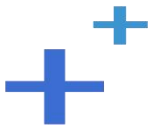
知识讲解



开发计划

知识讲解

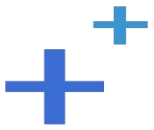
序号	内容	文档/代码	时间
26	声明全局变量	03_id/01_globals.h	3小时
27	定义全局变量	03_id/02_globals.cpp	
28	声明数据库访问类	03_id/03_db.h	
29	实现数据库访问类	03_id/04_db.cpp	
30	声明业务服务类	03_id/05_service.h	
31	实现业务服务类	03_id/06_service.cpp	



开发计划

知识讲解

序号	内容	文档/代码	时间
32	声明服务器类	03_id/07_server.h	3小时
33	实现服务器类	03_id/08_server.cpp	
34	定义主函数	03_id/09_main.cpp	
35	构建脚本	03_id/Makefile	
36	配置文件	etc/id.cfg	
37	建表脚本	sql/id.sql	



全局变量

配置信息

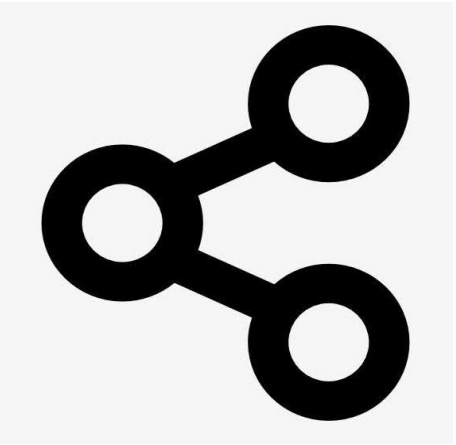
知识讲解

- 字符串配置表：cfg_str
  - MySQL地址表：cfg_maddrs
- 整型配置表：cfg_int
  - MySQL读写超时：cfg_mtimeout
  - 最大偏移：cfg_maxoffset

共享信息

知识讲解

- 配置共享信息
  - MySQL地址表：g_maddrs
- 其它共享信息
  - 主机名：g_hostname
  - ID表：g_ids
  - 互斥锁：g_mutex

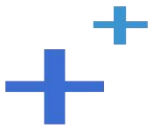


数据库访问类(db_c)

属性、构造和析构

知识讲解

- 成员变量
  - MySQL对象：m_mysql
- 构造函数：db_c
  - 创建MySQL对象
- 析构函数：~db_c
  - 销毁MySQL对象



方法

知识讲解

- 连接数据库：connect
  - 遍历MySQL地址表，尝试连接数据库
- 获取ID当前值，同时产生下一个值：get
  - 关闭自动提交
  - 查询数据库
  - 获取查询结果
  - 获取结果记录
    - 有记录：更新旧记录，提交数据库，输出当前值
    - 无记录：插入新记录，提交数据库，缺省当前值
  - 打开自动提交

id	id_value	create_time	update_time
../data	100	2020-11-01 12:55:03	2020-11-01 12:55:03

业务服务类(service_c)

一级方法

知识讲解

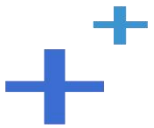
- 业务处理：business
  - 解析包头
    - 包体长度
    - 命令
    - 状态
  - 根据命令执行具体业务处理
    - 处理来自存储服务器的获取ID请求
  - 返回处理结果

二级方法

- 处理来自存储服务器的获取ID请求：get
  - 检查包体长度
  - 接收包体
  - 根据ID的键获取其值
  - 应答ID

知识讲解

包头			包体
包体长度	命令(40)	状态	ID键
8	1	1	64+1



三级方法

知识讲解

- 根据ID的键获取其值：get
  - 互斥锁加锁
  - 在ID表中查找ID
    - 找到该ID
      - 若该ID的偏移未及上限
        - 从内存中获取ID值
      - 否则
        - 从数据库中获取ID值
    - 未找到该ID
      - 从数据库中获取ID值
      - 添加该ID到ID表中
  - 互斥锁解锁

三级方法

- 从数据库中获取ID值：fromdb
  - 连接数据库
  - 获取ID当前值，同时产生下一个值

知识讲解

```
// 从数据库中获取ID值
long service_c::fromdb(char const* key) const {
    db_c db; // 数据库访问对象

    // 连接数据库
    if (db.connect() != OK)
        return -1;

    long value = -1;

    // 获取ID当前值，同时产生下一个值
    if (db.get(key, cfg_maxoffset, &value) != OK)
        return -1;

    return value;
}
```

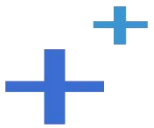
底层方法

知识讲解

- 应答ID：id
  - 构造响应
  - 发送响应
- 应答错误：error
  - 错误描述
  - 构造响应
  - 发送响应

包头			包体
包体长度	命令(101)	状态	ID值
8	1	1	8

包头			包体	
包体长度	命令(101)	状态	错误号	错误描述
8	1	1	2	<=1024

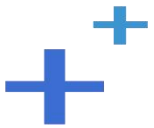


附录：程序清单

TNV/src/03_id/01_globals.h

知识讲解

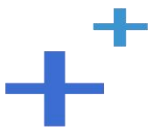
```
//ID服务器
//声明全局变量
//
#pragma once

#include <vector>
#include <lib_acl.hpp>
#include "01_types.h"
//
//配置信息
//
extern char* cfg_maddrs; //MySQL地址表
extern acl::master_str_tbl cfg_str[]; //字符串配置表
```



TNV/src/03_id/01_globals.h

```
extern int cfg_mtimeout;  // MySQL读写超时
extern int cfg_maxoffset; // 最大偏移
extern acl::master_int_tbl cfg_int[]; // 整型配置表

extern std::vector<std::string> g_maddrs; // MySQL地址表
extern std::string g_hostname; // 主机名
extern std::vector<id_pair_t> g_ids; // ID表
extern pthread_mutex_t g_mutex; // 互斥锁
```

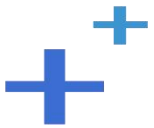
知识讲解



TNV/src/03_id/02_globals.cpp

知识讲解

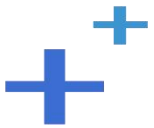
```
// ID服务器
// 定义全局变量
//
#include "01_globals.h"
//
// 配置信息
//
char* cfg_maddrs; // MySQL地址表
acl::master_str_tbl cfg_str[] = { // 字符串配置表
        {"mysql_addrs", "127.0.0.1", &cfg_maddrs},
        {0, 0, 0}};
```



TNV/src/03_id/02_globals.cpp

知识讲解

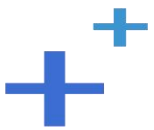
```
int cfg_mtimeout;  // MySQL读写超时
int cfg_maxoffset; // 最大偏移
acl::master_int_tbl cfg_int[] = { // 整型配置表
        {"mysql_rw_timeout",  30, &cfg_mtimeout,  0, 0},
        {"idinc_max_step",   100, &cfg_maxoffset, 0, 0},
        {0, 0, 0, 0, 0}};

std::vector<std::string> g_maddrs; // MySQL地址表
std::string g_hostname; // 主机名
std::vector<id_pair_t> g_ids; // ID表
pthread_mutex_t g_mutex = PTHREAD_MUTEX_INITIALIZER; // 互斥锁
```



TNV/src/03_id/03_db.h

知识讲解

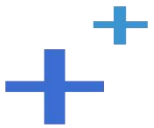
```
//ID服务器
//声明数据库访问类
//
#pragma once

#include <mysql.h>
//
//数据库访问类
//
class db_c {
public:
        //构造函数
        db_c(void);
```



TNV/src/03_id/03_db.h

知识讲解

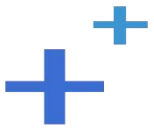
```
        //析构函数
        ~db_c(void);

        //连接数据库
        int connect(void);

        //获取ID当前值，同时产生下一个值
        int get(char const* key, int inc, long* value) const;

private:
        MYSQL* m_mysql; //MySQL对象
};
```



TNV/src/03_id/04_db.cpp

知识讲解

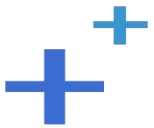
```
//ID服务器
//实现数据库访问类
//
#include "01_globals.h"
#include "03_db.h"

//构造函数
db_c::db_c(void): m_mysql(mysql_init(NULL)) { //创建MySQL对象
        if (!m_mysql)
                logger_error("create dao fail: %s", mysql_error(m_mysql));
}

//析构函数
db_c::~db_c(void) {
        //销毁MySQL对象
```



TNV/src/03_id/04_db.cpp

知识讲解

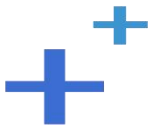
```
        if (m_mysql) {
                mysql_close(m_mysql);
                m_mysql = NULL;
        }
}

//连接数据库
int db_c::connect(void) {
        MYSQL* mysql = m_mysql;

        //遍历MySQL地址表，尝试连接数据库
        for (std::vector<std::string>::const_iterator maddr =
                g_maddrs.begin(); maddr != g_maddrs.end(); ++maddr)
                if ((m_mysql = mysql_real_connect(mysql, maddr->c_str(),
                        "root", "123456", "tnv_idsdb", 0, NULL, 0)))
```



TNV/src/03_id/04_db.cpp

知识讲解

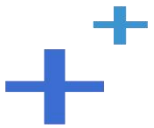
```
                return OK;

        logger_error("connect database fail: %s",
                mysql_error(m_mysql = mysql));
        return ERROR;
}

//获取ID当前值，同时产生下一个值
int db_c::get(char const* key, int inc, long* value) const {
        //关闭自动提交
        mysql_autocommit(m_mysql, 0);

        //查询数据库
        acl::string sql;
        sql.format("SELECT id_value FROM t_id_gen WHERE id='%s';", key);
```



TNV/src/03_id/04_db.cpp

知识讲解

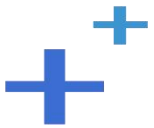
```
if (mysql_query(m_mysql, sql.c_str())) {
        logger_error("query database fail: %s, sql: %s",
                mysql_error(m_mysql), sql.c_str());
        mysql_autocommit(m_mysql, 1);
        return ERROR;
}

//获取查询结果
MYSQL_RES* res = mysql_store_result(m_mysql);
if (!res) {
        logger_error("result is null: %s, sql: %s",
                mysql_error(m_mysql), sql.c_str());
        mysql_autocommit(m_mysql, 1);
        return ERROR;
}
```



TNV/src/03_id/04_db.cpp

知识讲解

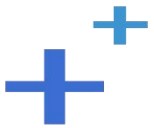
```
//获取结果记录
MYSQL_ROW row = mysql_fetch_row(res);
if (row) { //有记录
        //更新旧记录
        sql.format("UPDATE t_id_gen SET id_value="
                "id_value+%d WHERE id='%s';", inc, key);
        if (mysql_query(m_mysql, sql.c_str())) {
                logger_error("update database fail: %s, sql: %s",
                        mysql_error(m_mysql), sql.c_str());
                mysql_autocommit(m_mysql, 1);
                return ERROR;
        }
        //提交数据库
        mysql_commit(m_mysql);
        //库中当前值
```



TNV/src/03_id/04_db.cpp

知识讲解

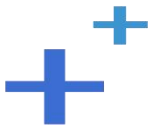
```
            *value = atol(row[0]);
        }
        else { //无记录
            //插入新记录
            sql.format("INSERT INTO t_id_gen SET id='%s', id_value='%d';",
                    key, inc);
            if (mysql_query(m_mysql, sql.c_str())) {
                logger_error("insert database fail: %s, sql: %s",
                        mysql_error(m_mysql), sql.c_str());
                mysql_autocommit(m_mysql, 1);
                return ERROR;
            }
            //提交数据库
            mysql_commit(m_mysql);
            //缺省当前值
```



TNV/src/03_id/04_db.cpp

```
            *value = 0;
        }

        //打开自动提交
        mysql_autocommit(m_mysql, 1);

        return OK;
    }
```

知识讲解



TNV/src/03_id/05_service.h

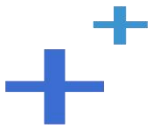
知识讲解

```
//ID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/03_id/05_service.h

知识讲解

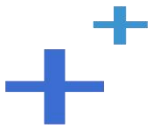
```
        //处理来自存储服务器的获取ID请求
        bool get(acl::socket_stream* conn, long long bodylen) const;

        //根据ID的键获取其值
        long get(char const* key) const;
        //从数据库中获取ID值
        long fromdb(char const* key) const;

        //应答ID
        bool id(acl::socket_stream* conn, long value) const;
        //应答错误
        bool error(acl::socket_stream* conn, short errnumb,
                char const* format, ...) const;
};
```



TNV/src/03_id/06_service.cpp

知识讲解

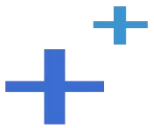
```
//ID服务器
//实现业务服务类
//
#include "02_proto.h"
#include "03_util.h"
#include "01_globals.h"
#include "03_db.h"
#include "05_service.h"

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



TNV/src/03_id/06_service.cpp

知识讲解

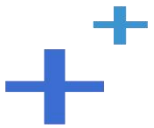
```
long long bodylen = ntoll(head); //包体长度
if (bodylen < 0) {
        error(conn, -1, "invalid body length: %lld < 0", bodylen);
        return false;
}
int command = head[BODYLEN_SIZE]; //命令
int status = head[BODYLEN_SIZE+COMMAND_SIZE]; //状态
logger("bodylen: %lld, command: %d, status: %d",
       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

bool result;

//根据命令执行具体业务处理
switch (command) {
        case CMD_ID_GET:
```



TNV/src/03_id/06_service.cpp

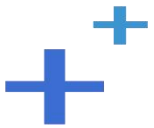
知识讲解

```
                        // 处理来自存储服务器的获取ID请求
                        result = get(conn, bodylen);
                        break;

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

        return result;
}

////////////////////////////////////////////////////////////////////////

// 处理来自存储服务器的获取ID请求
```



TNV/src/03_id/06_service.cpp

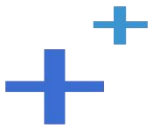

```
bool service_c::get(acl::socket_stream* conn, long long bodylen) const {
        //|包体长度|命令|状态|ID键|
        //|    8   |  1 |  1 |64+1|
        //检查包体长度
        long long expected = ID_KEY_MAX + 1; //期望包体长度
        if (bodylen > expected) {
                error(conn, -1, "invalid body length: %lld > %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",
```



TNV/src/03_id/06_service.cpp

知识讲解

```
                acl::last_serror(), bodylen, conn->get_peer());
        return false;
    }

    // 根据ID的键获取其值
    long value = get(body);
    if (value < 0) {
        error(conn, -1, "get id fail, key: %s", body);
        return false;
    }

    logger("get id ok, key: %s, value: %ld", body, value);

    return id(conn, value);
}
```



TNV/src/03_id/06_service.cpp

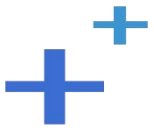
知识讲解

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

// 根据ID的键获取其值
long service_c::get(char const* key) const {
        // 互斥锁加锁
        if ((errno = pthread_mutex_lock(&g_mutex))) {
                logger_error("call pthread_mutex_lock fail: %s",
                        strerror(errno));
                return -1;
        }

        long value = -1;

        // 在ID表中查找ID
        std::vector<id_pair_t>::iterator id;
```

TNV/src/03_id/06_service.cpp

知识讲解

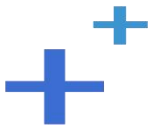
```
for (id = g_ids.begin(); id != g_ids.end(); ++id)
        if (!strcmp(id->id_key, key))
                break;
if (id != g_ids.end()) { //找到该ID
        if (id->id_offset < cfg_maxoffset) { //该ID的偏移未及上限
                value = id->id_value + id->id_offset;
                ++id->id_offset;
        }
        else if ((value = fromdb(key)) >= 0) { //从数据库中获取ID值
                //更新ID表中的ID
                id->id_value = value;
                id->id_offset = 1;
        }
}
else if ((value = fromdb(key)) >= 0) { //从数据库中获取ID值
```



TNV/src/03_id/06_service.cpp

知识讲解

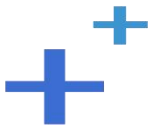
```
        // 在ID表中添加ID
        id_pair_t id;
        strcpy(id.id_key, key);
        id.id_value = value;
        id.id_offset = 1;
        g_ids.push_back(id);
}

// 互斥锁解锁
if ((errno = pthread_mutex_unlock(&g_mutex))) {
        logger_error("call pthread_mutex_unlock fail: %s",
                strerror(errno));
        return -1;
}
```



TNV/src/03_id/06_service.cpp

知识讲解

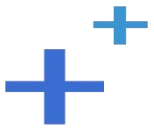
```
        return value;
}

//从数据库中获取ID值
long service_c::fromdb(char const* key) const {
        db_c db; //数据库访问对象

        //连接数据库
        if (db.connect() != OK)
                return -1;

        long value = -1;

        //获取ID当前值，同时产生下一个值
        if (db.get(key, cfg_maxoffset, &value) != OK)
```



TNV/src/03_id/06_service.cpp

知识讲解

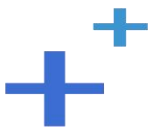
```
                return -1;

        return value;
}

////////////////////////////////////////////////////////////////////////

//应答ID
bool service_c::id(acl::socket_stream* conn, long value) const {
        //|包体长度|命令|状态|ID值|
        //|    8   |  1 |  1 |  8 |
        //构造响应
        long long bodylen = BODYLEN_SIZE;
        long long resplen = HEADLEN + bodylen;
        char resp[resplen] = {};
```



TNV/src/03_id/06_service.cpp

知识讲解

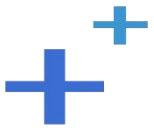
```
        llton(bodylen, resp);
        resp[BODYLEN_SIZE] = CMD_ID_REPLY;
        resp[BODYLEN_SIZE+COMMAND_SIZE] = 0;
        lton(value, resp + HEADLEN);

        //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/03_id/06_service.cpp

知识讲解

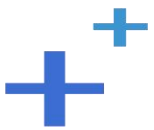
```
//应答错误
bool service_c::error(acl::socket_stream* conn, short errnumb,
        char const* format, ...) const {
        //错误描述
        char errdesc[ERROR_DESC_SIZE];
        va_list ap;
        va_start(ap, format);
        vsnprintf(errdesc, ERROR_DESC_SIZE, format, ap);
        va_end(ap);
        logger_error("%s", errdesc);
        acl::string desc;
        desc.format("[%s] %s", g_hostname.c_str(), errdesc);
        memset(errdesc, 0, sizeof(errdesc));
        strncpy(errdesc, desc.c_str(), ERROR_DESC_SIZE - 1);
        size_t desclen = strlen(errdesc);
```



TNV/src/03_id/06_service.cpp

知识讲解

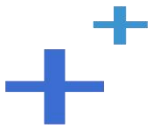
```
desclen += desclen != 0;

//|包体长度|命令|状态|错误号|错误描述|
//|    8   |  1 |  1 |   2  | <=1024 |
//构造响应
long long bodylen = ERROR_NUMB_SIZE + desclen;
long long resplen = HEADLEN + bodylen;
char resp[resplen] = {};
llton(bodylen, resp);
resp[BODYLEN_SIZE] = CMD_ID_REPLY;
resp[BODYLEN_SIZE+COMMAND_SIZE] = STATUS_ERROR;
ston(errnumb, resp + HEADLEN);
```



TNV/src/03_id/06_service.cpp

知识讲解

```
        if (desclen)
                strcpy(resp + HEADLEN + ERROR_NUMB_SIZE, errdesc);

        //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;
}
```



复习课见