Tedu.cn
达内教育

《分布式流媒体》实训项目

C/C++教学体系

TNV DAY10

直播课

目录

服务器类(server_c)

进程级回调方法

知识讲解

- 进程启动时被调用：proc_on_init
  - 检查并拆分MySQL地址表
  - 获取主机名
  - 检查最大偏移
  - 打印配置信息

```
// 进程切换用户后被调用
void server_c::proc_on_init(void) {
    // MySQL地址表
    if (!cfg_maddrs || !strlen(cfg_maddrs))
        logger_fatal("mysql addresses is null");
    split(cfg_maddrs, g_maddrs);
    if (g_maddrs.empty())
        logger_fatal("mysql addresses is empty");

    // 主机名
    char hostname[256+1] = {};
    if (gethostname(hostname, sizeof(hostname) - 1))
        logger_error("call gethostname fail: %s", strerror(errno));
    g_hostname = hostname;

    // 最大偏移不能太小
    if (cfg_maxoffset < 10)
        logger_fatal("invalid maximum offset: %d < 10", cfg_maxoffset);

    // 打印配置信息
    logger("cfg_maddrs: %s, cfg_mtimeout: %d, cfg_maxoffset: %d",
        cfg_maddrs, cfg_mtimeout, cfg_maxoffset);
}
```

进程级回调方法

- 进程意图退出时被调用：proc_exit_timer
  - 返回true，进程立即退出，否则若配置项ioctl_quick_abort非0，进程立即退出，否则待所有客户机连接都关闭后，进程再退出
  - 检查客户机数和客户线程数
    - 若其中至少有一个为零
      - 则立即退出
    - 否则
      - 待所有客户机连接都关闭后再退出，除非配置项ioctl_quick_abort非零

知识讲解

进程级回调方法

- 进程意图退出时被调用：proc_exit_timer

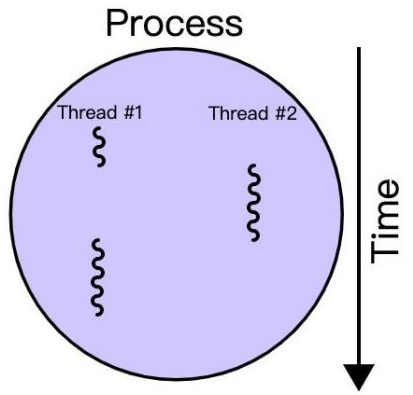
```
if (!nclients || !nthreads) {
        logger("nclients: %lu, nthreads: %lu", nclients, nthreads);
        return true;
}

return false;
```

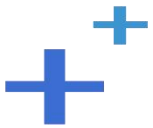
知识讲解

线程级回调方法

知识讲解

- 线程获得连接时被调用：thread_on_accept
  - 返回true，连接将被用于后续通信，否则函数返回后即关闭连接
  - 打印日志
- 线程连接可读时被调用：thread_on_read
  - 返回true，保持长连接，否则函数返回后即关闭连接
  - 接收包头
  - 业务处理

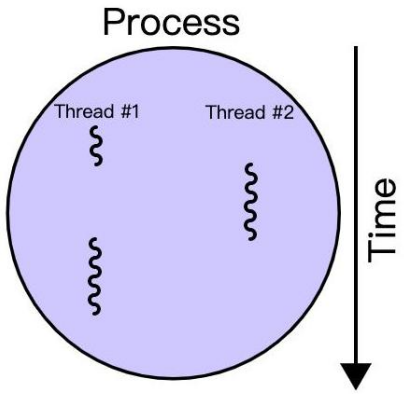




线程级回调方法

知识讲解

- 线程读写超时时被调用：thread_on_timeout
  - 返回true，继续等待下一次读写，否则函数返回后即关闭连接
  - 打印日志
  - 返回true以保持连接
- 线程连接关闭时被调用：thread_on_close
  - 打印日志

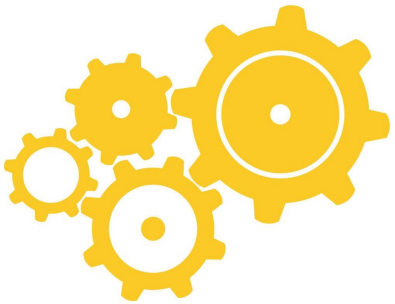


主函数(main)

主函数(main)

知识讲解

- 初始化ACL库
  - acl::acl_cpp_init();
  - acl::log::stdout_open(true);
- 创建并运行服务器
  - server_c& server = acl::singleton2<server_c>::get_instance();
  - server.set_cfg_str(cfg_str);
  - server.set_cfg_int(cfg_int);
  - server.run_alone("127.0.0.1:22000", "../etc/id.cfg");



构建脚本(Makefile)

构建脚本(Makefile)

知识讲解

```
PROJ = ../../bin/id
OBJS = $(patsubst %.cpp, %.o, $(wildcard ../01_common/*.cpp *.cpp))
CC = g++
LINK = g++
RM = rm -rf
CFLAGS = -c -Wall -I/usr/include/acl-lib/acl_cpp `mysql_config --
cflags` -I../01_common
LIBS = -pthread -lacl_all `mysql_config --libs`
```

构建脚本(Makefile)

知识讲解

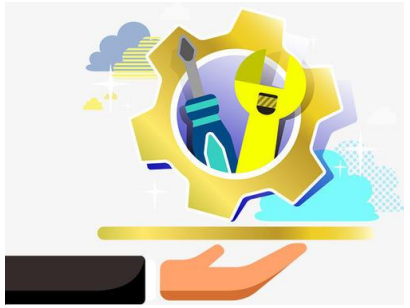
```
all: $(PROJ)
$(PROJ): $(OBJS)
        $(LINK) $^ $(LIBS) -o $@
.cpp.o:
        $(CC) $(CFLAGS) $^ -o $@
clean:
        $(RM) $(PROJ) $(OBJS)
```

配置文件(id.cfg)

配置文件(id.cfg)

- MySQL地址表：mysql_addrs = 127.0.0.1
- MySQL读写超时：mysql_rw_timeout = 30
- 最大偏移：idinc_max_step = 100

```
service id {
    # MySQL地址表
    mysql_addrs = 127.0.0.1
    # MySQL读写超时
    mysql_rw_timeout = 30
    # 最大偏移
    idinc_max_step = 100
}
```



建表脚本(id.sql)

建表脚本(id.sql)

```
USE tnv_idsdb
DROP TABLE IF EXISTS `t_id_gen`;
CREATE TABLE `t_id_gen` (
  `id` varchar(64) NOT NULL DEFAULT '',
  `id_value` bigint(20) DEFAULT NULL,
  `create_time` timestamp NULL DEFAULT CURRENT_TIMESTAMP,
  `update_time` timestamp NULL DEFAULT CURRENT_TIMESTAMP ON
  UPDATE CURRENT_TIMESTAMP,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```

附录：程序清单

TNV/src/03_id/07_server.h

知识讲解

```
//ID服务器
//声明服务器类
//
#pragma once

#include <lib_acl.hpp>
//
//服务器类
//
class server_c: public acl::master_threads {
protected:
        //进程切换用户后被调用
        void proc_on_init(void);
        //子进程意图退出时被调用
        //返回true，子进程立即退出，否则
        //若配置项ioctl_quick_abort非0，子进程立即退出，否则
        //待所有客户机连接都关闭后，子进程再退出
```

TNV/src/03_id/07_server.h

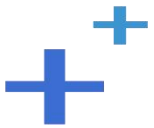
知识讲解

```
bool proc_exit_timer(size_t nclients, size_t nthreads);

//线程获得连接时被调用
//返回true，连接将被用于后续通信，否则
//函数返回后即关闭连接
bool thread_on_accept(acl::socket_stream* conn);
//与线程绑定的连接可读时被调用
//返回true，保持长连接，否则
//函数返回后即关闭连接
bool thread_on_read(acl::socket_stream* conn);
//线程读写连接超时时被调用
//返回true，继续等待下一次读写，否则
//函数返回后即关闭连接
bool thread_on_timeout(acl::socket_stream* conn);
//与线程绑定的连接关闭时被调用
void thread_on_close(acl::socket_stream* conn);
};
```

TNV/src/03_id/08_server.cpp

知识讲解

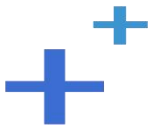
```
// ID服务器
// 实现服务器类
//
#include <unistd.h>
#include "02_proto.h"
#include "03_util.h"
#include "01_globals.h"
#include "05_service.h"
#include "07_server.h"

// 进程切换用户后被调用
void server_c::proc_on_init(void) {
        // MySQL地址表
        if (!cfg_maddrs || !strlen(cfg_maddrs))
                logger_fatal("mysql addresses is null");
```



TNV/src/03_id/08_server.cpp

知识讲解

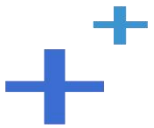
```
split(cfg_maddrs, g_maddrs);
if (g_maddrs.empty())
        logger_fatal("mysql addresses is empty");

//主机名
char hostname[256+1] = {};
if (gethostname(hostname, sizeof(hostname) - 1))
        logger_error("call gethostname fail: %s", strerror(errno));
g_hostname = hostname;

//最大偏移不能太小
if (cfg_maxoffset < 10)
        logger_fatal("invalid maximum offset: %d < 10", cfg_maxoffset);

//打印配置信息
```



TNV/src/03_id/08_server.cpp

```
        logger("cfg_maddrs: %s, cfg_mtimeout: %d, cfg_maxoffset: %d",
                cfg_maddrs, cfg_mtimeout, cfg_maxoffset);
}

//子进程意图退出时被调用
//返回true，子进程立即退出，否则
//若配置项ioctl_quick_abort非0，子进程立即退出，否则
//待所有客户机连接都关闭后，子进程再退出
bool server_c::proc_exit_timer(size_t nclients, size_t nthreads) {
        if (!nclients || !nthreads) {
                logger("nclients: %lu, nthreads: %lu", nclients, nthreads);
                return true;
        }

        return false;
```

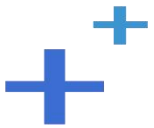
知识讲解



TNV/src/03_id/08_server.cpp

知识讲解

```
}

//线程获得连接时被调用
//返回true，连接将被用于后续通信，否则
//函数返回后即关闭连接
bool server_c::thread_on_accept(acl::socket_stream* conn) {
        logger("connect, from: %s", conn->get_peer());
        return true;
}

//与线程绑定的连接可读时被调用
//返回true，保持长连接，否则
//函数返回后即关闭连接
bool server_c::thread_on_read(acl::socket_stream* conn) {
        //接收包头
```



TNV/src/03_id/08_server.cpp

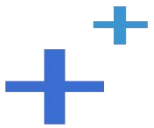
知识讲解

```
        char head[HEADLEN];
        if (conn->read(head, HEADLEN) < 0) {
                if (conn->eof())
                        logger("connection has been closed, from: %s",
                                conn->get_peer());
                else
                        logger_error("read fail: %s, from: %s",
                                acl::last_serror(), conn->get_peer());
                return false;
        }

        //业务处理
        service_c service;
        return service.business(conn, head);
}
```

TNV/src/03_id/08_server.cpp

知识讲解

```
//线程读写连接超时时被调用
//返回true，继续等待下一次读写，否则
//函数返回后即关闭连接
bool server_c::thread_on_timeout(acl::socket_stream* conn) {
        logger("read timeout, from: %s", conn->get_peer());
        return true;
}

//与线程绑定的连接关闭时被调用
void server_c::thread_on_close(acl::socket_stream* conn) {
        logger("client disconnect, from: %s", conn->get_peer());
}
```



TNV/src/03_id/09_main.cpp

知识讲解

```
//ID服务器
//定义主函数
//
#include "01_globals.h"
#include "07_server.h"

int main(void) {
        //初始化ACL库
        acl::acl_cpp_init();
        acl::log::stdout_open(true);
        //创建并运行服务器
        server_c& server = acl::singleton2<server_c>::get_instance();
        server.set_cfg_str(cfg_str);
        server.set_cfg_int(cfg_int);
        server.run_alone("127.0.0.1:22000", "../etc/id.cfg");
        return 0;
}
```

TNV/src/03_id/Makefile

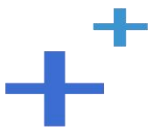
知识讲解

```
PROJ    = ../../bin/id
OBJS    = $(patsubst %.cpp, %.o, $(wildcard ../01_common/*.cpp *.cpp))
CC      = g++
LINK    = g++
RM      = rm -rf
CFLAGS = -c -Wall -I/usr/include/acl-lib/acl_cpp `mysql_config --cflags` -I../01_common
LIBS    = -pthread -lacl_all `mysql_config --libs`

all: $(PROJ)

$(PROJ): $(OBJS)
        $(LINK) $^ $(LIBS) -o $@
.cpp.o:
        $(CC) $(CFLAGS) $^ -o $@
clean:
        $(RM) $(PROJ) $(OBJS)
```

TNV/etc/id.cfg

```
service id {
        # MySQL地址表
        mysql_addrs = 127.0.0.1
        # MySQL读写超时
        mysql_rw_timeout = 30
        # 最大偏移
        idinc_max_step = 100
}
```

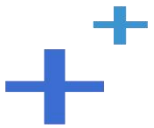
知识讲解



TNV/sql/id.sql

知识讲解

```sql
DROP DATABASE IF EXISTS tnv_idsdb;
CREATE DATABASE tnv_idsdb;
USE tnv_idsdb;

CREATE TABLE `t_id_gen` (
  `id` varchar(64) NOT NULL DEFAULT '',
  `id_value` bigint(20) DEFAULT NULL,
  `create_time` timestamp NULL DEFAULT CURRENT_TIMESTAMP,
  `update_time` timestamp NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```



复习课见