欢迎大家来到第五阶段课程

《分布式流媒体》实训项目

TNV DAY14

复习课

预习内容

HTTP服务器（6）

HTTP服务器（6）

服务器类(server_c)的进程级回调方法

知识讲解

- 进程启动时被调用：proc_on_init
  - 初始化客户机
  - 创建并初始化Redis集群
  - 打印配置信息
- 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

服务器类(server_c)的进程级回调方法

- 进程即将退出时被调用：proc_on_exit
  - 销毁Redis集群
  - 终结化客户机

知识讲解

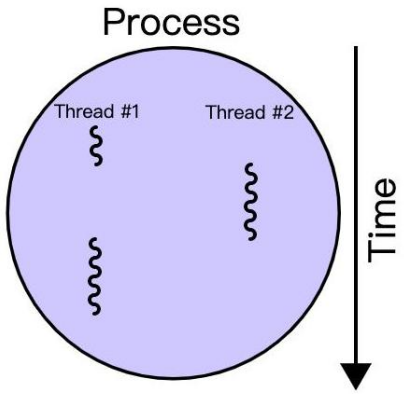
```
// 进程退出前被调用
void server_c::proc_on_exit(void) {
    delete m_redis;

    // 终结化客户机
    client_c::deinit();
}
```

服务器类(server_c)的线程级回调方法

知识讲解

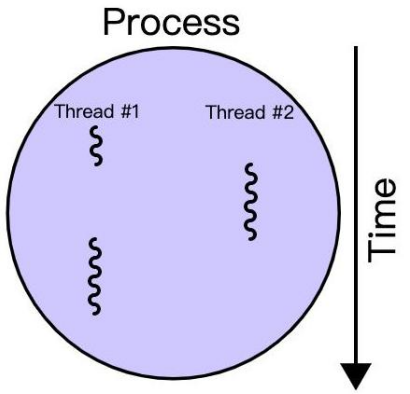
- 线程获得连接时被调用：thread_on_accept
  - 返回true，连接将被用于后续通信，否则函数返回后即关闭连接
  - 设置读写超时
  - 创建会话对象
  - 创建并设置业务服务对象
- 线程连接可读时被调用：thread_on_read
  - 返回true，保持长连接，否则函数返回后即关闭连接
  - 获取业务服务对象
  - 业务处理

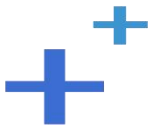


服务器类(server_c)的线程级回调方法

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false以关闭连接
- 线程连接关闭时被调用：thread_on_close
  - 销毁会话和业务服务对象





附录：程序清单

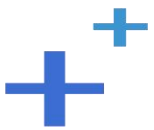
TNV/src/06_http/07_server.cpp

知识讲解

```
// HTTP服务器
// 实现服务器类
//
#include "01_types.h"
#include "07_client.h"
#include "02_globals.h"
#include "04_service.h"
#include "06_server.h"

// 进程切换用户后被调用
void server_c::proc_on_init(void) {
        // 初始化客户机
        client_c::init(cfg_taddrs);

        // 创建并初始化Redis集群
```



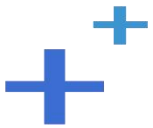
TNV/src/06_http/07_server.cpp

```
        m_redis = new acl::redis_client_cluster;
        m_redis->init(NULL, cfg_raddrs, cfg_maxthrds,
                cfg_ctimeout, cfg_rtimeout);

        //打印配置信息
        logger("cfg_taddrs: %s, cfg_raddrs: %s, cfg_maxthrds: %d, "
                "cfg_ctimeout: %d, cfg_rtimeout: %d, cfg_rsession: %d",
                cfg_taddrs, cfg_raddrs, cfg_maxthrds,
                cfg_ctimeout, cfg_rtimeout, cfg_rsession);
}

// 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
```

知识讲解



TNV/src/06_http/07_server.cpp

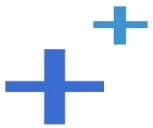
知识讲解

```
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;
}

//进程退出前被调用
void server_c::proc_on_exit(void) {
        delete m_redis;

        //终结化客户机
        client_c::deinit();
```



TNV/src/06_http/07_server.cpp

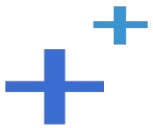
知识讲解

```
}

//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());

        //设置读写超时
        conn->set_rw_timeout(cfg_rtimeout);

        //创建会话
        acl::session* session = cfg_rsession ?
                (acl::session*)new acl::redis_session(*m_redis, cfg_maxthrds) :
                (acl::session*)new acl::memcache_session("127.0.0.1:11211");
```



TNV/src/06_http/07_server.cpp

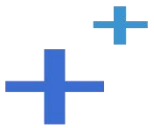
知识讲解

```
        //创建并设置业务服务对象
        conn->set_ctx(new service_c(conn, session));

        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) {
        service_c* service = (service_c*)conn->get_ctx();
        if (!service)
                logger_fatal("service is null");

        return service->doRun();
```



TNV/src/06_http/07_server.cpp


```
}

//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 false;
}

//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());

        service_c* service = (service_c*)conn->get_ctx();
        acl::session* session = &service->getSession();
        delete session;
        delete service;
}
```

下节课见