ACL Master 框架文档

概述

ACL Master 框架是 acl_cpp 库中提供的服务器框架模块，用于简化网络服务器的开发。该框架提供了多种服务器模型，支持在 acl_master 守护进程管理下运行，也支持独立运行模式。

目录结构

主要组件

1. master_base

基础抽象类，所有 master 服务器类的基类。提供了配置管理、生命周期回调等基础功能。

头文件: `acl_cpp/master/master_base.hpp`

2. master_threads

多线程服务器模型，基于线程池处理客户端连接。

头文件: `acl_cpp/master/master_threads.hpp`

特点:

- 支持线程池管理
- 每个连接由一个线程处理
- 适用于长连接场景
- 支持异步读写控制

3. master_proc

多进程服务器模型，每个连接由独立进程处理。

头文件: `acl_cpp/master/master_proc.hpp`

特点:

- 进程隔离，稳定性高
- 适用于处理时间较长的请求
- 单连接单进程模式

4. master_aio

异步 I/O 服务器模型，基于事件驱动的异步处理。

头文件: `acl_cpp/master/master_aio.hpp`

特点:

- 基于事件驱动
- 高并发处理能力
- 支持多种事件引擎（select、poll、epoll、kqueue 等）
- 适用于高并发短连接场景

5. master_trigger

定时触发服务器模型，按固定间隔执行任务。

头文件: `acl_cpp/master/master_trigger.hpp`

特点:

- 定时触发执行
- 不处理网络连接
- 适用于定时任务场景

6. master_udp

UDP 服务器模型，处理 UDP 数据包。

头文件: `acl_cpp/master/master_udp.hpp`

特点:

- 支持 UDP 协议
- 基于线程池处理
- 支持多个 UDP 监听地址

7. master_fiber

协程服务器模型，基于协程（fiber）处理客户端连接。

头文件: `fiber/master_fiber.hpp`

特点:

- 基于协程，轻量级并发
- 同步编程模型，异步执行
- 极高的并发处理能力
- 内存占用少，上下文切换快
- 适用于高并发场景

8. master_conf

配置管理类，用于加载和管理服务器配置。

头文件: `acl_cpp/master/master_conf.hpp`

特点:

- 支持多种配置类型（bool、int、int64、string）
- 配置文件加载
- 配置参数验证

运行模式

Daemon 模式

在 acl_master 守护进程管理下运行，通过配置文件控制服务参数：

```
int main(int argc, char* argv[]) {
    master_threads server;
    server.run_daemon(argc, argv);
    return 0;
}
```

独立模式

独立运行，不依赖 acl_master，适用于调试和测试：

```
int main() {
    master_threads server;
    server.run_alone("127.0.0.1:8080", "./test.cf");
    return 0;
}
```

生命周期回调

Master 框架提供了完整的生命周期回调机制：

回调方法	说明	调用时机
proc_on_listen	监听成功回调	每个监听地址绑定成功后
proc_pre_jail	切换用户前回调	进程切换用户权限之前
proc_on_init	初始化回调	进程/线程初始化完成后
proc_pre_exit	退出前回调	进程退出前，可控制是否退出
proc_on_exit	退出回调	进程最终退出前
proc_on_sighup	信号回调	收到 SIGHUP 信号时

配置参数类型

框架支持四种配置参数类型：

master_bool_tbl

布尔型配置参数：

```
master_bool_tbl bool_tab[] = {
    { "enable_feature", 1, &g_enable_feature },
    { NULL, 0, NULL }
};
```

master_int_tbl

整型配置参数：

```
master_int_tbl int_tab[] = {
    { "io_timeout", 120, &g_io_timeout, 10, 600 },
    { NULL, 0, NULL, 0, 0 }
};
```

master_int64_tbl

64位整型配置参数：

```
master_int64_tbl int64_tab[] = {
    { "max_size", 1000000, &g_max_size, 0, 10000000 },
    { NULL, 0, NULL, 0, 0 }
};
```

master_str_tbl

字符串配置参数：

```
master_str_tbl str_tab[] = {
    { "log_file", "/var/log/server.log", &g_log_file },
    { NULL, NULL, NULL }
};
```

选择合适的服务器模型

场景	推荐模型	原因
超高并发	master_fiber	协程模型，极高并发能力
高并发短连接	master_aio / master_fiber	异步处理，资源占用少

场景	推荐模型	原因
长连接，中等并发	master_threads	线程池处理，编程简单
对稳定性要求高	master_proc	进程隔离，互不影响
UDP 协议	master_udp	专门的 UDP 支持
定时任务	master_trigger	定时触发机制

依赖关系

```
master_base (基类)
    ├── master_conf (配置管理)
    ├── master_threads (线程模型)
    ├── master_proc (进程模型)
    ├── master_aio (异步模型)
    ├── master_trigger (触发器模型)
    ├── master_udp (UDP模型)
    └── master_fiber (协程模型)
```

编译要求

- 仅在非 `ACL_CLIENT_ONLY` 模式下可用
- 需要 C++11 或更高版本
- 依赖 acl 和 acl_cpp 库

相关文档

- ACL 官方文档
- ACL 示例代码

许可证

遵循 ACL 项目的许可证