**功能**：Linux上实现基于TCP和Unix domain Socket的本地IPC通信模型。

**结构：**采用CSS(client-server-service)模式，一个server负责管理client和service服务。可以在server上注册多个service，service是可以自定义的，一个Client在connet到server时可以指定绑定到某个service，这个service可以为client提供服务。一个service可以绑定多个client，一个client在运行的过程中也可以重绑定到其他service。Client和server建立的是Unix domain socket链接，log接受器和server建立的是TCP链接。

框架：

Server

Log Service

B Service

A Service

Log receiver

r

Clientn

Client3

Client2

Client1

Client0

目录说明：

Server: server核心代码所在目录

Services: service服务代码，用户可以自己添加service服务

Log\_recv: log接收器，可以在filters数组上添加过滤字段，过滤自己感兴趣的log。凡是通过Log.h头文件定义的LOG\_OUTPUT输出的log都可以用Log\_recv接收。

Test: 测试例子

Client请求的消息模式：

1. 带超时等待callback返回的请求
2. 不需要等待callback返回的请求
3. 异步执行callback的请求
4. 不需要callback的请求

Message.h头文件中定义的结构体：

typedef struct

{

call\_back cb\_func;

int cb\_async;

int wait\_time; // // -1: wait forever, 0: not wait, > 0: wait ms

}stCbInfo;

\* 如果cb\_func要执行比较耗时的操作将影响service服务器处理其他client的请求

\* 因此需要将cb\_async置为1, 使cb\_func在client\_connect的时候

\* 创建的线程中执行，这样可以避免service被一个client占用而影响其他client.

\*

\* 如果cb\_async被置为0则可以分为两种情况：

（1） 如果wait\_time 大于0 则client 需要等待callback执行完才能继续，且超时时间为wait\_time 毫秒。

（2） 如果wait\_time 等于0， 则client向服务器发送完命令后，不需要等待可直接执行

环境搭建：

(1)下载 waf工具：<https://waf.io/> or $ curl -o waf-2.0.9.tar.bz2 <https://waf.io/waf-2.0.9.tar.bz2>

(2) 解压

$ tar xjvf waf-2.0.9.tar.bz2

$ cd waf-2.0.9

$ ./waf-light

$ cd waf-2.0.9

$ PATH=$PATH:$PWD

(3) 编译

$ waf configure build

(4) 运行

$ cd IPC\_Model

$ cp ./build/server/libserver.so /usr/lib

或者export LD\_LIBRARY\_PATH=$PWD/build/server

$ ./build/test/server\_test