

基于 c 技术环境的微信公众号框架引擎 (weixin4c)

Calvin

目录索引

1	背景	4
2	架构与设计	4
2.1	体系架构	4
2.2	系统架构	5
2.3	应用结构	5
3	安装和配置	8
4	开发示例	8
4.1	从模板部署应用和配置	8
4.2	修改配置	9
4.3	编译安装空应用架子	9
4.4	开发应用	9
4.4.1	修改 main 函数	9
4.4.2	编写公众号消息回调函数	11
4.4.3	部署应用	11
4.5	获取接口调用凭据	11
5	使用参考	12
5.1	weixin4c 引擎	12
5.1.1	struct Weixin4cConfig 结构	12
5.1.2	类型消息回调函数	12
5.2	weixin4c 公共层	12
5.2.1	网址	13
5.2.2	HTTP 协议	13
5.2.3	字符编码转换	13
5.2.4	文件	14
5.2.5	工具	14
6	日志文件	14

版本修订

版本号	修订日期	修订人	修订内容
0.2.0	2016-01-10	Calvin	创建文档

1 背景

weixin4c 是一个基于 C 技术环境的建立在 FastCGI 高性能 Web 应用服务器架构上的微信公众号框架引擎，它封装了公众号应用服务器底层框架，你只需要编写业务逻辑代码，以回调函数机制接入框架，就能快速构造出微信公众号服务。

用 C 技术环境开发微信公众号应用的确有些奇怪，但是我没有互联网公司高大上的团队配置（俺只熟悉 C）、也没有雄厚的资金支持购买强劲的服務器（俺只有一台单核、半 G 内存、1M 带宽的阿里云最低配主机），所以只能利用最熟悉的 C、利用最有限的硬件资源来玩玩微信公众号了，如果选择 JAVA 估计连跑起来都成问题。

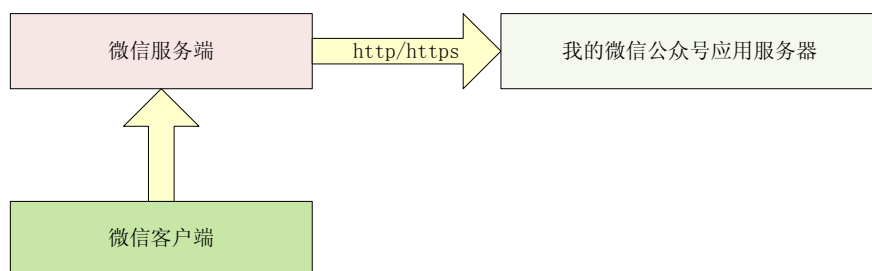
无论用什么语言，首先都要评估所要做的项目的技术背景和技术原理，只要是开放接口都与实现无关，采用 C 语言，虽然没有大量现成的库可以直接调用，但经过评估，自己研发成本并不大。之前我专门研究了 whois 爬虫技术，可以把这个作为第一个功能提供给用户使用。于是开搞！

元旦花了一天研究了微信公众平台开发者文档，撰写测试代码以评估接口，确定设计，又花了一天做开发测试，“钛搜索”新鲜上线，第一版提供了域名注册信息实时查询，而后某个工作日晚上又把框架部分和应用部分分离，把框架提炼出来封装成 C 库，方便做其它公众号，同时也分享给大家研究使用。

2 架构与设计

2.1 体系架构

通过学习微信公众平台开发者文档，我的微信公众号应用服务器接入架构应该是这样的



其中微信客户端可以是手机、PC 等，微信服务端其实只做了验证我的应用服务器地址，然后大量转发客户端请求功能，对于我要做的只是自己找网上主机空间搭建一个 **Web** 服务器，根据开放接口开发相应服务对接腾讯微信服务端即可。

幸运的是，前段时间我刚好买了一个阿里云主机，虽然是低的不能再低的配置（没钱啊），但跑跑 C 程序是毫无问题。

2.2 系统架构

既然采用 C 技术环境，CGI 是板上钉钉，但传统的 CGI 存在频繁创建、销毁进程的系统压力，那就使用 **FastCGI** 吧。

本人曾经研发过基于事件模型(**epoll** in **Linux**)的服务器软件，感觉要完全写好事件模型还是比较复杂，于是对 **Nginx** 的稳定性感觉不是很有信心，还是采用老而弥坚的 **Apache**。至于你的选择，只要你的 **Web** 服务器软件支持 **FastCGI** 就能使用 **weixin4c**。

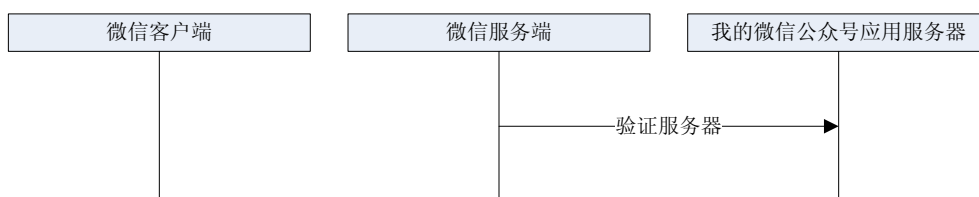
暂时还用不到数据库或缓存服务器，以后用到了再加。

总结：支持 **FastCGI** 的 **Web** 服务器软件+**FastCGI(C)**模块+(你的应用).**fcgi** (**weixin4c.a**)。（这恐怕是世界上执行效率最高的基于高级语言的 **Web** 技术方案了吧，呵呵呵，开发效率么...我有大量自己写的、第三方的库，不比 **JAVA**、**PHP** 开发慢哦 ^_^)

2.3 应用结构

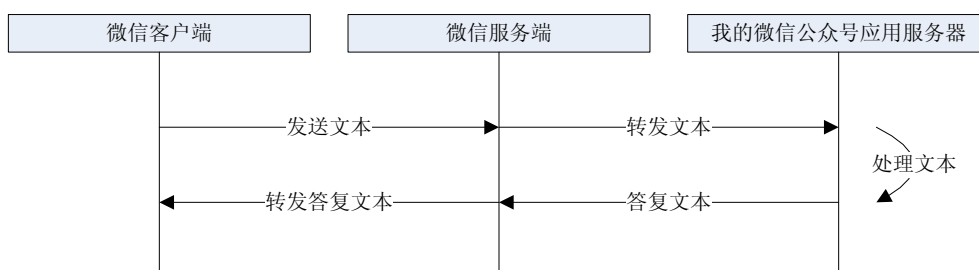
公众号应用服务器对接微信服务器的第一步是让微信服务器认证自己的网

址是否有效，具体参见微信公众平台开发文档

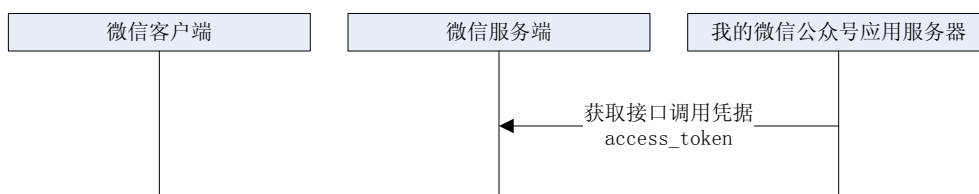


一旦认证成功，你的应用服务就算接入微信了，接下去就可以互相收发信息。

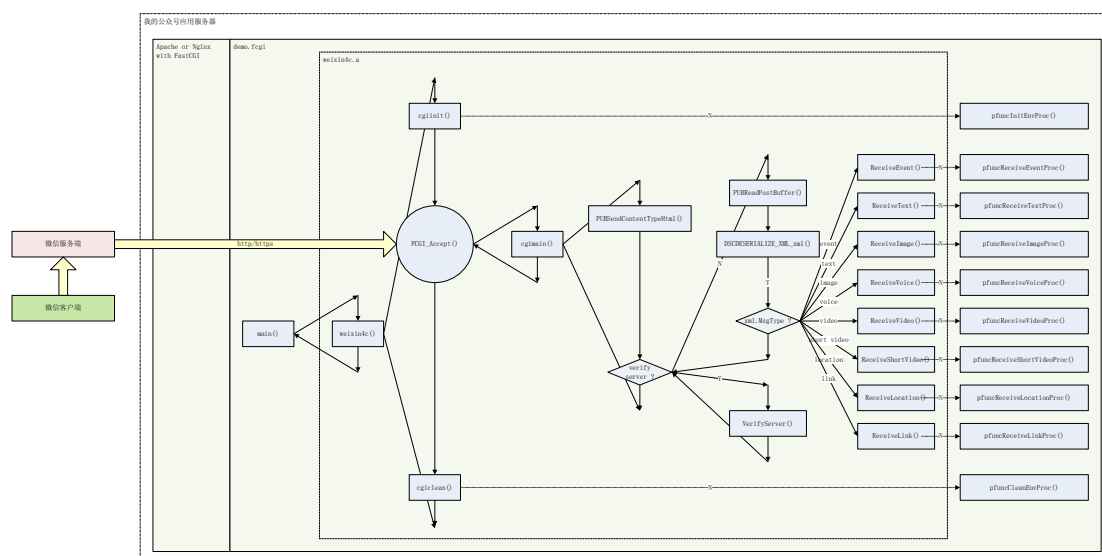
比较常见的是微信客户端上发送文本、图片、语音、视频、地理位置信息等，微信服务器转发给你的应用服务器，应用服务器处理后产生响应信息，回复给微信客户端



还有一些是公众号应用服务器主动发送微信服务器，微信服务器转发给微信客户端，由于安全原因，微信服务器需要公众号应用服务器事先通过获取的接口调用凭据来验证发送者身份



这种模式放到后面再说，前一种模式的 weixin4c 内部设计结构如下：



(在 word 里显示有点小, 你可以直接看源码安装包里的"doc/weixin4c.vsd")

Web 服务器进程根据微信服务端转发的请求, 调起应用 `demo.fcgi` (你可以改成"(你的公众号应用名).fcgi"), `main` 函数里设置一些环境常量和回调函数指针, 然后调用 `weixin4c.a` 入口函数 `weixin4c`, `weixin4c` 前半部分实现了一个 FastCGI 基本框架, 后半部分实现了微信公众号请求信息接收、筛选、分支调用的通用层框架, 公众号应用处理逻辑由 `demo.fcgi` 回调函数来挂接实现。

目前实现的回调函数接口有:

- 进程初始化函数 (FastCGI 长进程启动时执行一次, 像打开 `redis` 连接、打开数据库连接啥啥的都可以塞这里面)
- 进程销毁前清理函数 (FastCGI 长进程结束时执行一次, 其实应用不释放也会被系统释放, 但有些应用层正常关闭操作最好还是做一下, 如断开数据库连接)
- 接收事件 (订阅/关注、取消订阅/关注) 处理函数
- 接收文本处理函数 (当微信客户端发送消息时触发, 可以用作命令的响应)
- 接收图片处理函数 (以下有些响应报文布局可能会有些问题)
- 接收语音处理函数
- 接收视频处理函数
- 接收小视频处理函数 (以上有些响应报文布局可能会有些问题)
- 接收地理信息处理函数
- 接收链接处理函数

备注: 为什么采用静态库 `libweixin4c.a` 而不采用动态库呢? 因为如果采用动态库, 需要设置一大堆 CGI 环境变量, 干脆就采用静态库, 这样减少框架层运行时库依赖, 方便使用。

总结: "`weixin4c.a`"只是实现了一个微信公众号分拣引擎, 由 `cgi` 可执行程序 "(你的应用名).fcgi"中的 `main` 函数调用, `main` 函数同时还会传递一批回调函数指针进去, 供"`weixin4c.a`"分拣引擎确定请求报文类型后回调, 就酱紫, 很简单吧 ^_^

3 安装和配置

从 git 托管网站上搜索下载 weixin4c 源码包或 ZIP 包，比如在 git.oschina.net、github.com。

放到你的环境里展开，这里假设你的环境是 Linux

如果是源码包，需要先编译安装

```
$ cd src
$ vi makeinstall
```

默认编译安装到本用户下

- 可执行文件(access token)安装到\$HOME/bin/
- 头文件(weixin4c.h、...)安装到\$HOME/include/weixin4c/
- 库文件(libweixin4c_public.a、libweixin4c.a)安装到\$HOME/lib/

你可以修改_HDERBASE、_LIBBASE、_BINBASE 或_HDERINST、_LIBINST、_BININST 以编译安装到其它目录

```
$ make -f makefile.Linux clean && make -f makefile.Linux install
```

如果没有报错，恭喜您，编译安装成功 ^_^

4 开发示例

4.1 从模板部署应用和配置

以我的“钛搜索”(taisousuo)公众号应用开发为例，你的项目应该改成你的名字

复制配置文件和应用代码模板到你的目录里

```
$ cd ..
$ cp -rf etc ~/etc/taisousuo
$ cp -rf demo ~/src/taisousuo
```


4.2 修改配置

进入\$HOME/etc/taisousuo，修改你的微信公众号配置，其中 AccessToken 不用改，后面用定时刷新接口调用凭据程序 accesstoken 来写。

4.3 编译安装空应用架子

进入\$HOME/src/taisousuo，修改 makefile.Linux 里的 FastCGI 可执行文件名从 demo 到 taisousuo

```
BIN          =      demo.fcgi
和
demo.fcgi    :      $(C_FILE_O)
```

默认编译安装目录为\$HOME/www/cgi-bin，如果你的不同目录，则修改 makeinstall

安装包自带 demo 是个可编译的空应用架子，下面开始编译部署应用

```
$ make -f makefile.Linux clean && make -f makefile.Linux install
```

如果没有报错，恭喜您，编译部署应用成功，你可以在安装目录 \$HOME/www/cgi-bin/里找到刚才编译部署出来的给腾讯微信服务端调用的 CGI 程序。

4.4 开发应用

4.4.1 修改 main 函数

```
$ cd $HOME/src/taisousuo
$ vi main.c
```

安装包自带 demo 里的 main.c 长这个样子

```
#include "weixin4c.h"

int main()
{
    struct Weixin4cConfig  conf ;
```

```

memset( & conf , 0x00 , sizeof(conf) );
conf.run_mode = WEIXIN4C_RUNMODE_PRODUCT ;
conf.home = "/home/demo" ;
conf.project_name = "demo" ;
conf.funcs.pfuncReceiveEventProc = & ReceiveEventProc ;
conf.funcs.pfuncReceiveTextProc = & ReceiveTextProc ;
conf.funcs.pfuncReceiveImageProc = & ReceiveImageProc ;
conf.funcs.pfuncReceiveVoiceProc = & ReceiveVoiceProc ;
conf.funcs.pfuncReceiveVideoProc = & ReceiveVideoProc ;
conf.funcs.pfuncReceiveShortVideoProc = &
ReceiveShortVideoProc ;
conf.funcs.pfuncReceiveLocationProc = & ReceiveLocationProc ;
conf.funcs.pfuncReceiveLinkProc = & ReceiveLinkProc ;

return -weixin4c( & conf );
}

```

也就是填充一个 `weixin4c` 配置结构体，然后传递给微信公众号消息分拣引擎入口函数 `weixin4c`。

配置结构体成员 `run_mode` 设置了运行模式，生产模式 `WEIXIN4C_RUNMODE_PRODUCT` 则完全以 `FastCGI` 长进程运行，性能较高，调试模式 `WEIXIN4C_RUNMODE_DEBUG` 则以 `CGI` 短进程运行，每次重新编译部署应用后无需重启 `Web` 服务器软件即可马上测试。

配置结构体成员 `home` 指向系统用户主目录，用于创建日志文件等。

配置结构体成员 `project_name` 指向项目名（微信公众号英文名），用于读配置文件（`$HOME/etc/项目名/AppID`）等。

配置结构体成员 `funcs` 下面函数指针用于挂接需要处理的消息类型的回调函数，`weixin4c` 引擎在分拣消息时，如果你配置了某个类型的消息回调函数，则运行之，如果没有配置回调函数，则返回空消息。如果想在用户订阅后马上回复一句话，需要设置事件类型消息回调 `pfuncReceiveEventProc`。如果只是提供接收命令请求、发送命令处理响应的話，需要设置文本类型消息回调 `pfuncReceiveTextProc`。暂时用不到的回调函数指针可以设置成 `NULL` 或者干脆删掉赋值行。

4.4.2 编写公众号消息回调函数

根据 `main` 函数里设置的回调，填充相应函数内容（源代码文件名与函数名同名），如事件类型消息回调函数 `ReceiveEventProc` 在 `ReceiveEventProc.c`，如文本类型消息回调函数 `ReceiveTextProc` 在 `ReceiveTextProc.c`。

4.4.3 部署应用

开发完后重新编译部署应用

```
$ make -f makefile.Linux clean && make -f makefile.Linux install
```

如果当前是生产模式，如果重启 `Web` 服务器软件；如果当前是调试模式，可以直接在微信客户端测试。

4.5 获取接口调用凭据

`weixin4c` 自带了一个获取接口调用凭据守护程序，用于定期自动向微信服务器发起获取凭据，并保存在本地配置文件中（默认在 `$HOME/etc/` 项目名 `/AccessToken`），供公众号应用服务器主动发起请求时使用。

启动命令为：（以“钛搜索”为例）

```
$ nohup accesstoken taisousuo 1 &
```

启动后，`accesstoken` 就以守护进程运行，关闭终端也不会受到影响，直到你 `kill` 它。

5 使用参考

5.1 weixin4c 引擎

5.1.1 struct Weixin4cConfig 结构

（见 4.4.1 修改 main 函数 章节）

5.1.2 类型消息回调函数

如文本类型消息回调函数原型，一般函数原型为

```
int ReceiveTextProc( void *user_data , xml *p_req , char *output_buffer , int *p_output_buflen , int *p_output_bufsize );
```

`void *user_data` 根据需要在 main 函数里设置到 conf 中，可以在 InitEnvProc 函数里填充，在 CleanEnvProc 函数里清理，在各大回调函数中使用，如定义一个应用私有结构体。

`xml *p_req` 引擎解包好请求报文后分拣给各回调函数使用，结构定义见 IDL_xml.dsc.h。原始数据中有些字段有 CDATA 包裹，引擎内已经调用公共函数 TakeoffCDATA 去掉了。

`char *output_buffer` 输出缓冲区基地址

`int *p_output_buflen` 输出缓冲区内有效数据长度

`int *p_output_bufsize` 输出缓冲区实际申请大小（暂时还不能改变，留给将来的版本扩展）

5.2 weixin4c 公共层

weixin4c 公共层包含了大量函数用于 weixin4c 框架引擎，也可以被用户应用层调用。

头文件默认安装在 \$HOME/include/weixin4c/weixin4c_public.h

库文件爱你默认安装在\$HOME/lib/libweixin4c_public.a

5.2.1 网址

得到原始的网址参数值地址指针

```
int PUBGetUrlParamPtr( char *key , char **pp_value , int *p_value_len );
```

得到原始的网址参数值，复制到应用给的缓冲区中

```
int PUBGetUrlParam( char *key , char *value , int value_size );
```

得到原始的网址参数值，复制到内部申请的内存块中，应用用完后应用负责释放

```
int PUBDupUrlParam( char *key , char **pp_value );
```

5.2.2 HTTP 协议

得到原始的 POST 数据缓冲区基地址

```
char *PUBGetPostBufferPtr();
```

得到原始的 POST 数据缓冲区有效数据长度

```
int PUBGetPostBufferLength();
```

5.2.3 字符编码转换

字符编码转换低层函数

```
int PUBConvCharacterCodeEx( char *from_character_code , char *in_buf , int in_len , char *to_character_code , char *out_buf , int out_len );
```

字符编码转换高层函数

```
int PUBConvCharacterCode( char *from_character_code , char *to_character_code , char *buf , int len , int size );
```

转换输出到内部申请的内存块中，应用用完后应用负责释放

```
int PUBDupConvCharacterCode( char *from_character_code , char *to_character_code , char *buf , int len , char **out_dup );
```

5.2.4 文件

读文件内容到内部申请的内存块中，应用用完后应用负责释放

```
int PUBReadEntireFileSafely( char *filename , char *mode , char
**pp_buf , long *p_file_size );
```

写文件内容

```
int PUBWriteEntireFile( char *filename , char *mode , char *p_buf ,
long file_size );
```

5.2.5 工具

取较小值宏

```
#define MAX( _a_, _b_ ) ( _a_ > _b_ ? _a_ : _b_ )
```

取较大值宏

```
#define MIN( _a_, _b_ ) ( _a_ < _b_ ? _a_ : _b_ )
```

用于清除字符串右边的换行符

```
char *PUBStringNoEnter( char *str );
```

用于字符串十六进制展开

```
int PUBHexExpand( char *HexBuf , int HexBufLen , char *AscBuf );
```

用于脱掉字符串两边的 CDATA，如果有的话

```
void TakeoffCDATA( char *str_with_cdata );
```

6 日志文件

基于 weixin4c 框架的进程在运行时会产生日志文件，便于分析运行情况。

日志目录为\$HOME/log，下列是各日志文件说明：

(项目名)_weixin4c.log weixin4c 函数及框架引擎写的日志

(项目名)_ReceiveText.log 文本类型消息回调函数里写的日志，其它类型消息回调函数日志同

(项目名)_accesstoken.log accesstoken 进程写的日志