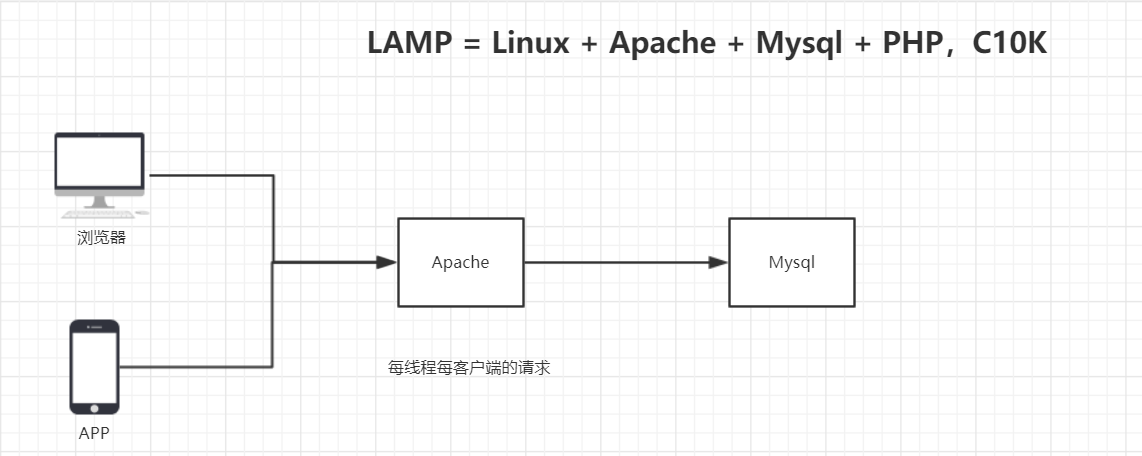
libevent

多路复用，

long long ago

LAMP,



===》 C10K

多路复用：

select poll epoll kqueue /dev/poll

1. 跨平台开发，移植开发：可移植性（跨平台）网络库，ACE : 单纯，学术和声望 : C++一些设计模式。 Module，陈硕，用之者死，学之者生。 libevent：libuv

* 创建事件集

struct event\_base \*event\_base\_new(void);

* 创建事件

struct event *event\_new(struct event\_base* ,evutil\_socket\_t fd, short EV\_READ | EV\_WRITE,event\_callback\_fn,void\* argv)

typedef void (\*event\_callback\_fn)(evutil\_socket\_t, short, void \* argv);

* 添加事件

int event\_add(struct event \* ev,const struct timeval\* timeout)

* 删除事件

int event\_del(struct event \*)

* 事件循环

int event\_base\_loop(struct event\_base \*base, int flags)

int event\_base\_dispatch(struct event\_base \*event\_base)

对各个平台调用socket的接口进行封装，提供统一的API供用户使用。

socket的事件回调函数一定是这么写的：

typedef void (\*event\_callback\_fn)(evutil\_socket\_t, short, void \* argv);

程序的最后调用event\_base\_dispatch(base);实现事件的循环处理

while（）

{

调用多路复用

……

}

1. 创建socket
2. 创建事件集event\_base
3. 创建event(socket, EV\_READ, callback1) / event(socket, EV\_WRITE, callback2)
4. 被event添加到事件集event\_base
5. event\_base\_dispatch(evnet\_base); event\_base\_loop();

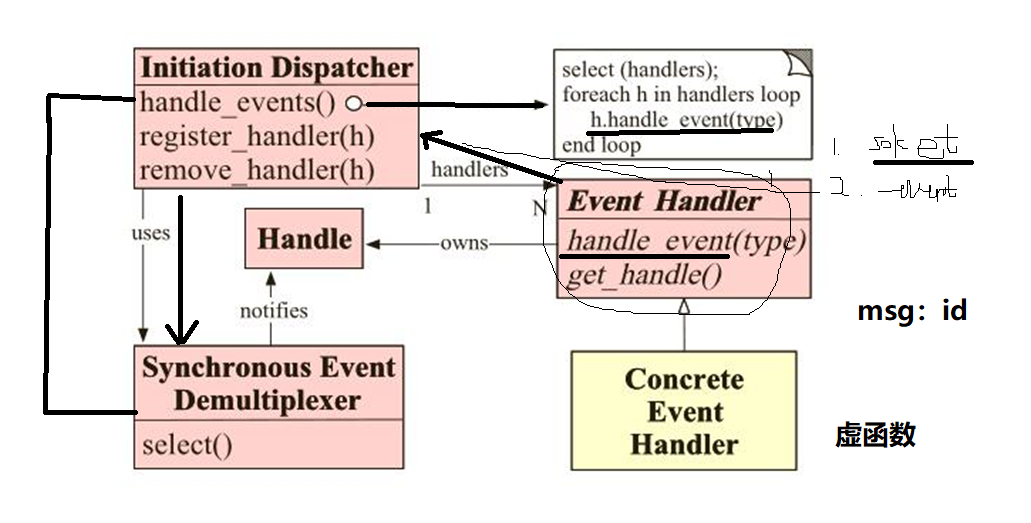
libevent还封装了http和https，定时器，信号。

README：安装 wiki，doc，sample。

你如果做后台开发的，想要装装B的，那就必须去了解几个设计模式，提高设计能力：

1. 订阅发布机制（事件驱动的设计）
2. 还有一个反应堆模式（线程池结合）
3. 工厂模式；
4. 过滤器模式；
5. 桥接模式：第三方库

实现设计：





表格驱动式：对于select、poll和epoll进行了抽象，抽象出了eventop的结构，这个结构定义如下：

struct eventop {

/\*\* The name of this backend. \*/

const char \*name;

void \*(\*init)(struct event\_base \*);

int (\*add)(struct event\_base \*, evutil\_socket\_t fd, short old, short events, void \*fdinfo);

int (\*del)(struct event\_base \*, evutil\_socket\_t fd, short old, short events, void \*fdinfo);

int (\*dispatch)(struct event\_base \*, struct timeval \*);

void (\*dealloc)(struct event\_base \*);

int need\_reinit;

enum event\_method\_feature features;

size\_t fdinfo\_len;

};

把这些所有的东西都存储在event.c文件里的eventops

/\* Array of backends in order of preference. \*/

static const struct eventop \*eventops[] = {

#ifdef EVENT\_\_HAVE\_EVENT\_PORTS

&evportops,

#endif

…

};

反应堆仅仅是触发EV\_READ / EV\_WRITE消息，而这个消息的处理确实丢给我们应用程序自己去处理（回调函数）。 对于epoll来讲，是一种异步化的处理。