**Skynet框架之菜鸟手册**

**一个轻量级的网络游戏服务器**

**Ver 0.1.1**

**wangdali <wangdali@qq.com>**

**2014-04**

目录

[一、 简介 4](#_Toc386706548)

[二、环境 4](#_Toc386706549)

[1、VMware下载与安装 4](#_Toc386706550)

[2、Ubuntu下载与安装 5](#_Toc386706551)

[3、Ubuntu 环境配置 6](#_Toc386706552)

[三、下载 9](#_Toc386706553)

[1、git使用 9](#_Toc386706554)

[2、github.com 12](#_Toc386706555)

[四、编译 13](#_Toc386706556)

[1、Makefile介绍 13](#_Toc386706557)

[2、核心程序介绍 17](#_Toc386706558)

[五、例子 20](#_Toc386706559)

[1、运行例子 20](#_Toc386706560)

[2、代码分析 22](#_Toc386706561)

[六、原理 26](#_Toc386706562)

[1、启动流程 26](#_Toc386706563)

[2、调用服务 27](#_Toc386706564)

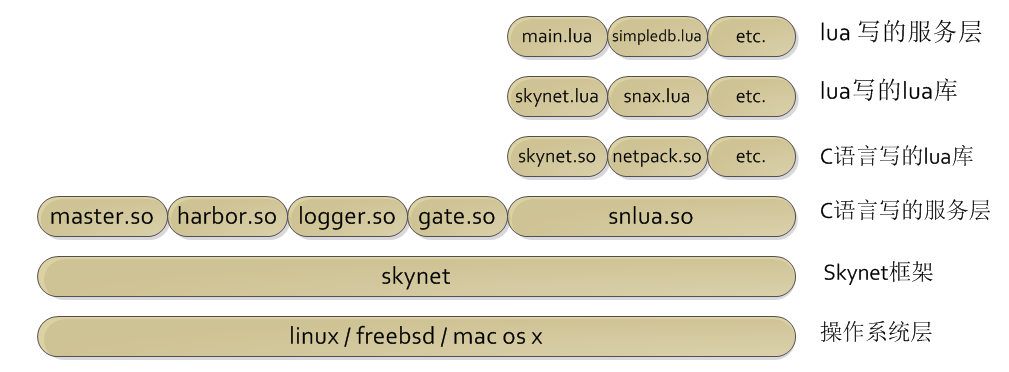
[七、服务 27](#_Toc386706565)

[1、用C语言写一个服务 27](#_Toc386706566)

[2、用Lua语言写一个服务 28](#_Toc386706567)

# 简介

Skynet是一个轻量级的网络服务器架构，并不是一个完整的游戏服务端。是服务端的最底层框架，和游戏有关的各种服务都是基于架构之上开发的。所以别想着拿Skynet改改就能作为网络游戏（包括私服），跑起来，甚至怎么监听客户端连接的服务都要你自己来写。Skynet的功能只有管理好服务（加载与调度）和服务之间的调用（请求与响应）。Skynet今后大部分都要在lua脚本下开发，只有需要考虑性能的模块才用C语言开发成lua库，提供给lua调用。现在Skynet提供给lua使用的库还不多，期望以后这些库多了，可以方便、简单的完成一个网络游戏服务端的开发。



# 二、环境

## 1、VMware下载与安装

我们既然是菜鸟，就在虚拟中安装Linux好了。首先我们得先选择一款虚拟机软件，有微软的Virtual PC 这个是免费的；甲骨文的VirtualBox也是免费的；我们选择最常用的VMware好了，这有两个版本，一个叫VMware Player，这个是免费的，可以在官方网站<http://www.vmware.com> 中下载，另一个叫VMware Workstation，这个是收费的。以上的几个随便选一个都可以用，不过我们菜鸟还是不差钱，就选收费的VMware Workstation好了。

安装VMware Workstation要写好多页纸，这里咋们就环保了吧，不懂安装的请找谷歌或者百度帮忙。现在VMware Workstation最高版本是10版。我们就选择这个了。

## 2、Ubuntu下载与安装

Skynet目前可以在以下操作系统上编译：1、Linux； 2、FreeBSD； 3、Mac OS X下编译使用。需要提醒的是不能在咱们菜鸟最喜欢的Windows下编译使用，虽然有老鸟在干这个事，但暂时没有可用的版本。

咱菜鸟还是选择Linux下使用好了，这个网上能查到的资料多，不懂得都能搜索到。既然决定用Linux我们就选择一个发行版简单点，常用的发行版有：1、RedHat红帽这个很有名，咱就不用了，你非要用我也不管，后面的内容可能会不适用；2、CentOS这个就是RedHat，不介绍了；3、Ubuntu这个可是十大受欢迎发行版之首。咱就选它好了，简单易用。

Ubuntu很友好的给咱们国内定制了一个版本，叫Kylin，中文叫麒麟。下载地址: <http://www.ubuntu.com/desktop/ubuntu-kylin-zh-CN>

根据你的Windows版本选择麒麟的版本，有32位和64位之分。32位的Windows下不能装64位版本的麒麟。所以请看在windows “计算机”处右键“属性”查看版本。



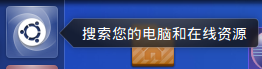
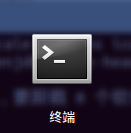
我的是“64位版操作系统”，对应下载ubuntukylin-14.04-desktop-amd64.iso 。

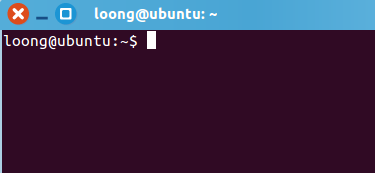
在VMware中安装Ubuntu的文章网上很多，而且安装过程很简单，这里就不叙述了。

## 3、Ubuntu 环境配置

Ubuntu安装好之后，还需要安装一些程序才能编译Skynet，所以我们先在这里配置一下，第一步我们需要启动一个命令行的“终端”，用命令来安装简单一点。

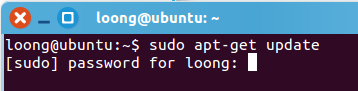
因为VMware自动安装的原因，我装完是英文的，你可以把它设置成中文。 里面找到设置用户的语言为汉语

首先我们得找到“终端”并启动它。我们启动Ubuntu在虚拟机中屏幕的左上角会看到这个图标： 请点击一下，新安装这里面没有曾经启动和使用过的文件列表。我们需要选择： 第二个，这个时候会看到 这里面就是所有你已经安装的软件列表了，相当于Window的“开始”—〉“程序”。上面提示还有76个结果没有显示出来，点一下把他们都显示出来。找到 这个就是我们需要的终端了。然后屏幕左边的菜单可以看到 右键点击一下，把“终端”“锁定到启动器”这样以后就不用到里面找“终端”了，相当Windows把图标锁定到任务栏。桌面上还可以看到“终端”的窗口，类似于：



好了，我们可以回到正题了，首先我们把Ubuntu更新到最新版本，在“终端”输入：

$ sudo apt-get update



意思是叫你输入loong这个用户的密码（我安装的时候起名叫loong，意为龙，你可以用别的名字，一定要记住密码），用来启动超级用户的权限，相当于Windows的“以管理员身份运行”。输入密码等待自动更新软件列表完成，接着就是更新软件了。在“终端”输入：

$ sudo apt-get upgrade



等待软件更新完成。虽然可以同时开多个“终端”但不能同时用来更新和安装软件，咱菜鸟们还是老老实实的等它们完成吧。如果出现错误和无法下载、无法解析之类的提示，就说明你的虚拟机没有联网，自己搞定它，可以用 firefox 火狐浏览器上网看看。

安装Skynet需要的软件：1、autoconf 准确的说这个是Skynet下使用的子模块jemalloc 编译的时候需要的。2、libreadline-dev。输入下面命令。

$ sudo apt-get install autoconf

$ sudo apt-get install libreadline-dev

执行命令的过程中可能会提示你需要下载多大的程序，安装需要多少空间，你是否继续，回答Y就好了。等待装完吧。

# 三、下载

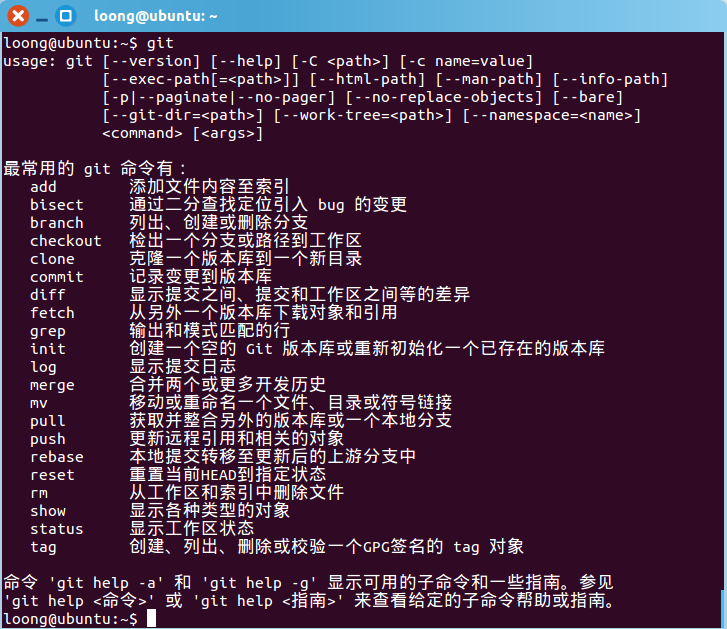
## 1、git使用

因为Skynet使用git来管理源代码的版本，所以最好还是简单的学习一下git。默认的Ubuntu没有安装git，我们需要自己装一下，打开“终端”输入：

$ sudo apt-get install git

输入git看下使用方法

$ git

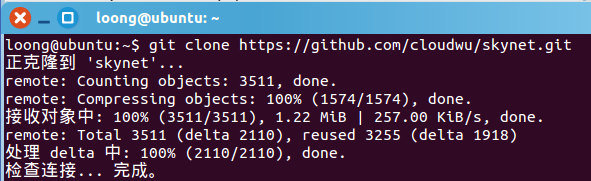


对于下载源代码，我们只需要知道“clone 克隆一个版本库到一个新目录”这一条怎么使用就好了。

接下来我们开始下载Skynet的源代码：

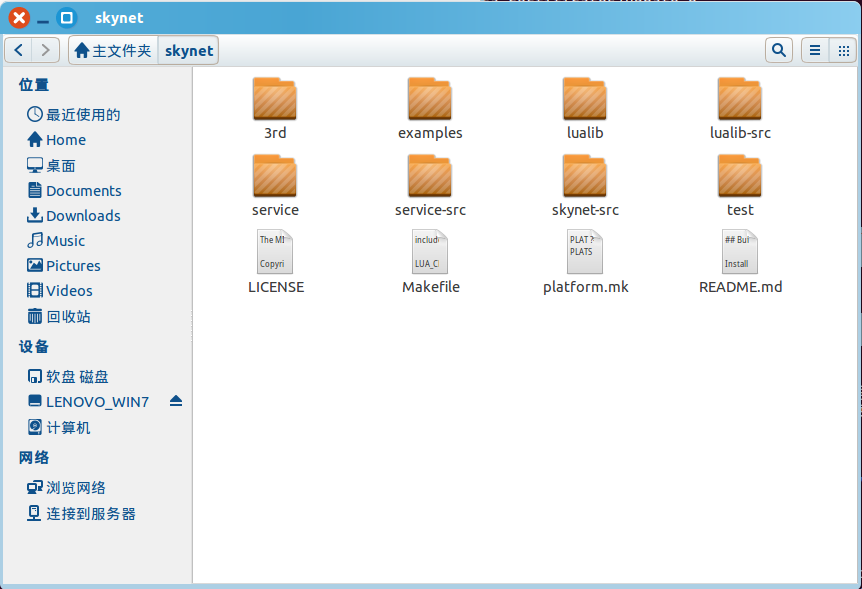
$ cd ~

$git clone <https://github.com/cloudwu/skynet.git>



到目前为止我们已经把Skynet的源代码下载到了

/home/loong/skynet 目录下：



上面就是源代码目录的内容，我们就简单的介绍一下目录里面都有什么东西吧：

README.md 简单介绍了怎么编译和测试Skynet

LICENSE 许可证信息，采用MIT，很宽松的协议。

Makefile 编译规则文件，用于编译Skynet

platform.mk 编译与平台相关的设置

3rd 第三方的代码，有lua和jemalloc等

examples 附带的例子

lualib 使用lua写的库

lualib-src 使用C写并封装给lua使用的库

service 使用lua写的Skynet的服务模块

service-src 使用C写的Skynet的服务模块

skynet-src Skynet的核心代码

test 使用lua写的一些测试代码

以上的核心代码为：skynet-src目录和service-src目录中的代码。剩下的基本都是为了给lua脚本使用的。对于我们菜鸟根本不用关心C语言写的东西。完全在lua上开发就好了。底层的东西留给云风吧。

3rd 中的jemalloc是一个内存分配的库，用来提高malloc的性能，默认情况下在linux下是开启使用的，如果你不想使用可以在platform.mk文件中添加下面语句屏蔽掉：

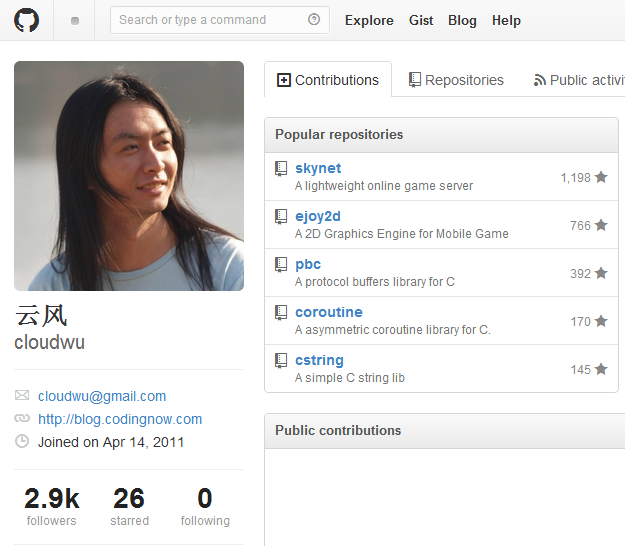
linux : SKYNET\_DEFINES :=-DNOUSE\_JEMALLOC

3rd 中的lua是云风修改过的，你也可以使用官方版本，修改的内容看这里：http://lua-users.org/lists/lua-l/2014-03/msg00489.html

## 2、github.com

Skynet的源代码托管在github.com下，你可以访问：

<https://github.com/cloudwu> 看到云风的其他开源代码，比如ejoy2d游戏引擎等。



八卦下，云风现在创业忙，瘦了，还是以前这张帅。要多注意身体呀。

点击 有更多的开源项目，在此谢谢云风。

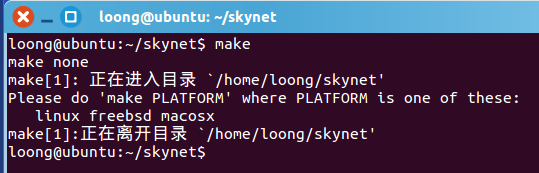
# 四、编译

## 1、Makefile介绍

唠叨了半天终于到这一步了。本手册是面对菜鸟的，没办法说了一大堆没用的东西。

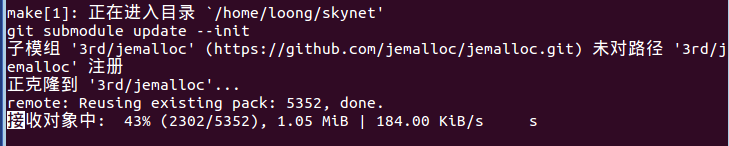
在linux一般都是使用Makefile来管理项目的，相当于我们在windows下使用Visual Studo 的 .sln 。linux编译很简单，一般就是:

$ make

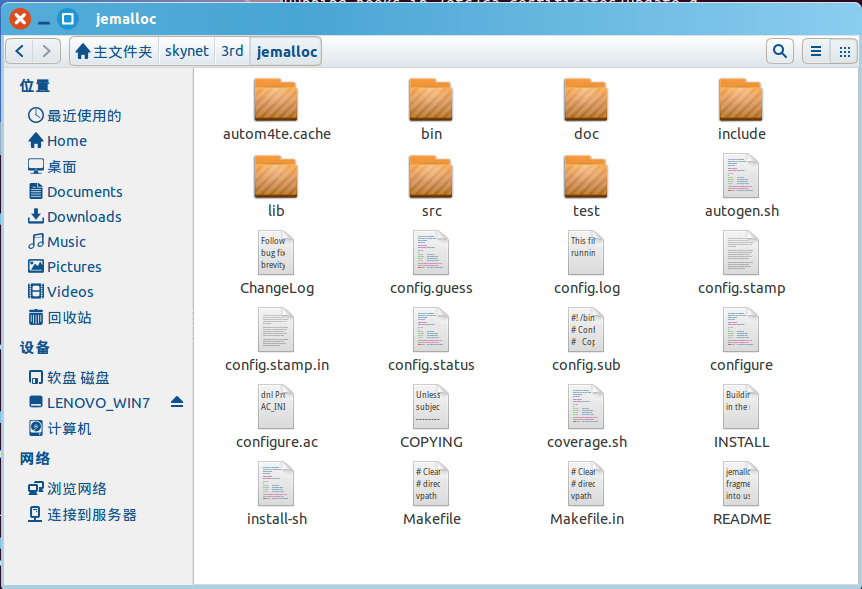


意思是你make的时候要带有平台名，因为我们是在linux下：

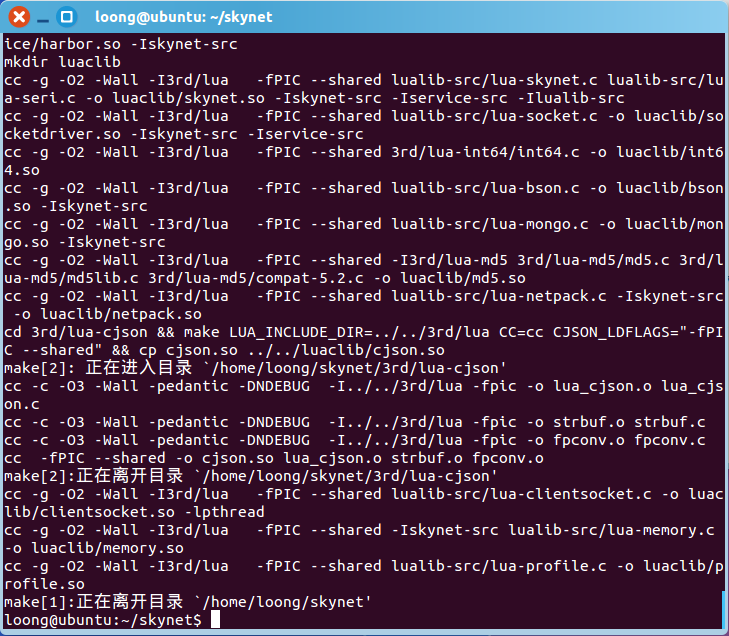
$make linux



会自动的去下载git submoudle 的子项目，即3rd/jemalloc 目录中的文件，刚开始这个目录是空的，现在有内容了。



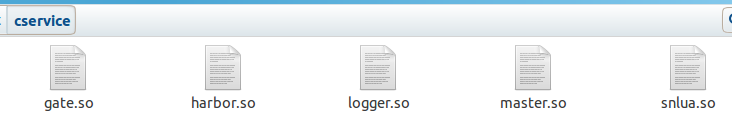
编译成功后会看到：



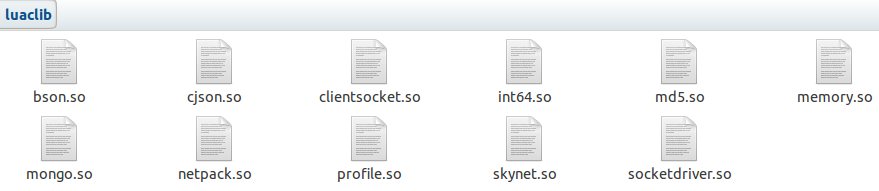
看看编译完后skynet目录下都多出了那些程序：

这个是主程序，linux没有后缀名，相当windows下的.exe文件。

多了个cservice目录，这个目录里面就是用 C 语言写的Skynet的服务模块，即service-src编译后的动态链接库。我们看看都有哪些。



多了个luaclib目录，这个目录里面就是用C 语言写的提供给lua脚本使用的库，即lualib-src编译后的动态链接库。



后面我们再介绍它的作用。

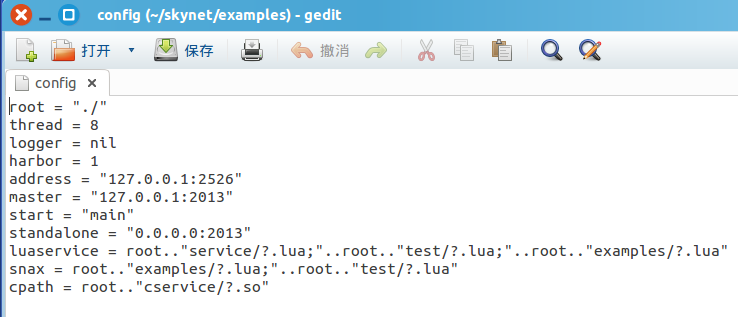
编译的其他东西我们就不关心了，在3rd 目录里面。这里的md5.so、int64.so、cjson.so就是3rd里面编译出来的。

## 2、核心程序介绍

①主程序，架构从这里启动，可在“终端”中输入：

$ ./skynet examples/config

./说明是当前目录，skynet是程序名，examples/config 为配置文件的路径和名字，这里的配置文件是用lua 写的一个脚本。内容为：



这里最重要的一条就是：start=”main” 意思就是设置第一个启动的服务。这里指向examples/main.lua 脚本服务模块。因为下面配置的luaservice的路径，所以不需要我们再输入examples/了，skynet会在配置的目录下找到main.lua脚本。cpath是用C语言写的服务模块路径。root=”./” 表示根目录为skynet启动的目录。thread为启动的工作线程数。logger=nil表示不记录日志，你可以指定一个路径和文件名，这样 “终端”输出的内容就写到日志中文件了。

其中standalone =”0.0.0.0:2013” 这个是master服务监听的地址和端口；address 和 master是节点使用的，address是节点的监听地址和端口，master是节点要连接的master服务的地址。

②主服务，就做两件事，回应名字的查询和在更新名字后，同步到其他节点。其他节点在启动时会把自己的harbor注册到master服务中。这样透过master服务就能把各个节点都连在一起，相互沟通了。每个节点之间都相互建立一条双向的通讯通道。

③节点服务，每个Skynet运行都是一个节点，在examples/main.lua中使用harbor来配置节点的编号，Skynet限制只有1到255个节点（保留0给系统内部使用），并把harbor的8bit值放在服务Id的最高8bit，因为服务Id是32bit的，所以Skynet的本地服务只能用低24位来表示，一个Skynet最多可以启动16M个服务模块。

④日志服务，一个简单的日志系统，可以用来记录服务的相关信息。

⑤网关服务，管理Socket。

⑥lua服务，用于加载lua写的skynet服务模块，而不仅仅限制于C语言才能写服务模块。十分核心和重要的模块。当然如果你不喜欢lua，你也可以换成python等其他脚本引擎。

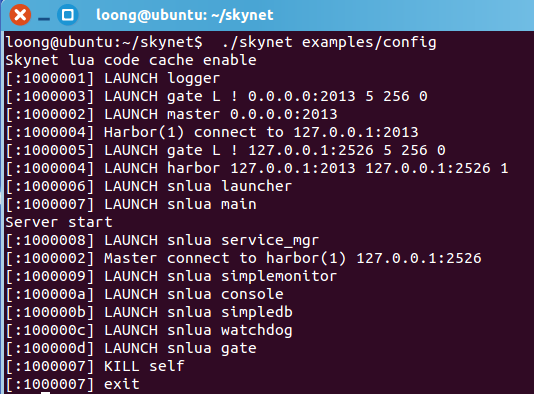
# 五、例子

## 1、运行例子

前面也说了，Skynet带有一个例子，放在examples目录下，这个例子就是实现一个简单的功能，类似于memcached。一个内存Key – Value系统。你可以通过指令 set key value来设置一个key和 value，他们是相互对应的关系，然后可以通过get key来获得上面设置的value的这个值。

我们来启动这个例子看看：

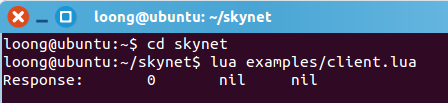
$ ./skynet examples/config



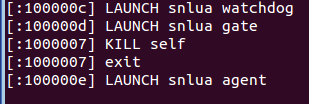
启动成功，至于[:1000007]KILL self自杀是因为 [:1000007]LAUNCH snlua main 即main.lua服务已经完成了它的任务，它自己在脚本中设置退出了。

客户端，我们使用examples/client.lua即可。启动它

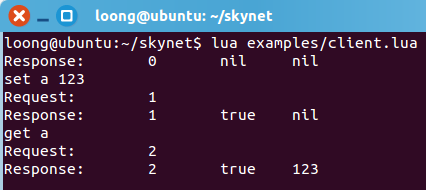
$ lua examples/client.lua



服务端会看到有客户端链接进来：



每一个客户端都会启动一个agent处理它的请求。我们简单使用一下：



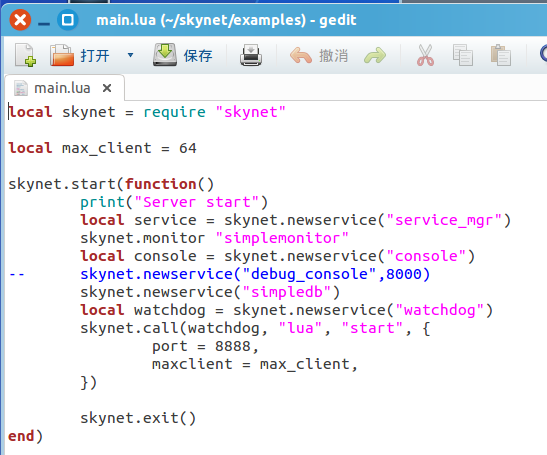
输入：set a 123

然后：get a

即，先设置a=123，然后试下get a是不是获得a的值为123。

## 2、代码分析

Skynet 启动examples/config 脚本，通过start=”main” 开启了第一个Skynet的服务main.lua，这个文件也在examples中，我们看看这个文件写的什么：



第一行先引用skynet这个库，这个是用lua写的，即lualib/skynet.lua。里面定义了这些接口：

skynet.start()用于服务的入口，加载lua服务时先运行这里的代码，它调用了luaclib-src/lua\_skynet.c里面的callback()，最终调用Skynet的框架skynet\_callback()来设置回调函数。

skynet.newservice() 用于启动一个lua写的服务，省略掉.lua后缀名。它调用了skynet.call() 然后skynet.call()调用luaclib-src/lua\_skynet.c里面的send()，最终调用Skynet的框架skynet\_send()压入队列。

skynet.call()用于发送一条消息给Skynet的框架。消息会压入队列，等待Skynet框架的调度。

skynet.exit()移除服务，通过skynet.send()发送一条消息给Skynet框架来移除lua的这个服务。

skynet.monitor() 用于监视服务，看它是否关闭。

main.lua一共打开了四个服务：

1. service\_mgr 这个是系统的模块，用于管理服务。
2. console 这个是系统的模块，用于输出。
3. simpledb 这个是例子的模块，用于管理Key – Value数据。
4. watchdog 这个是例子的模块，用于监视socket端口，等待数据。

main.lua没有调用其它函数，加载完服务，它也就完成了任务，所以它最后调用了skynet.exit()把自己杀掉了。

现在Skynet已经启动了watchdog服务，监听着8888端口，等待客户端的链接。

下面是watchdog服务的skynet\_start()开始函数：



skynet.dispatch()这个服务的回调函数，通过SOCKET[]来调用函数，这些函数有：

SOCKET.open() 打开 agent服务并启动，使用gate来管理socket。

SOCKET.close() 关闭 agent服务。

SOCKET.error() 打印错误信息。

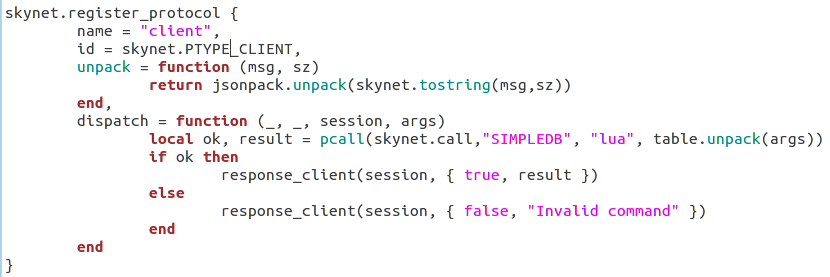
SOCKET.data() 有数据到来。

下面就来看看agent服务的代码：



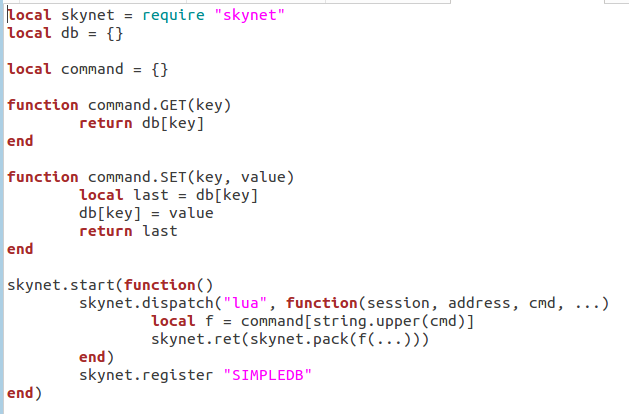
前面watchdog调用SOCKET.open()的时候就调用了这里的CMD.start()，在客户端输出了”Welcome to skynet”。

Agent的核心就是注册了协议，并根据协议把数据发送给simpledb服务去处理：



协议的详细部分看lualib/skynet.lua。

最后我们看看simpledb服务：

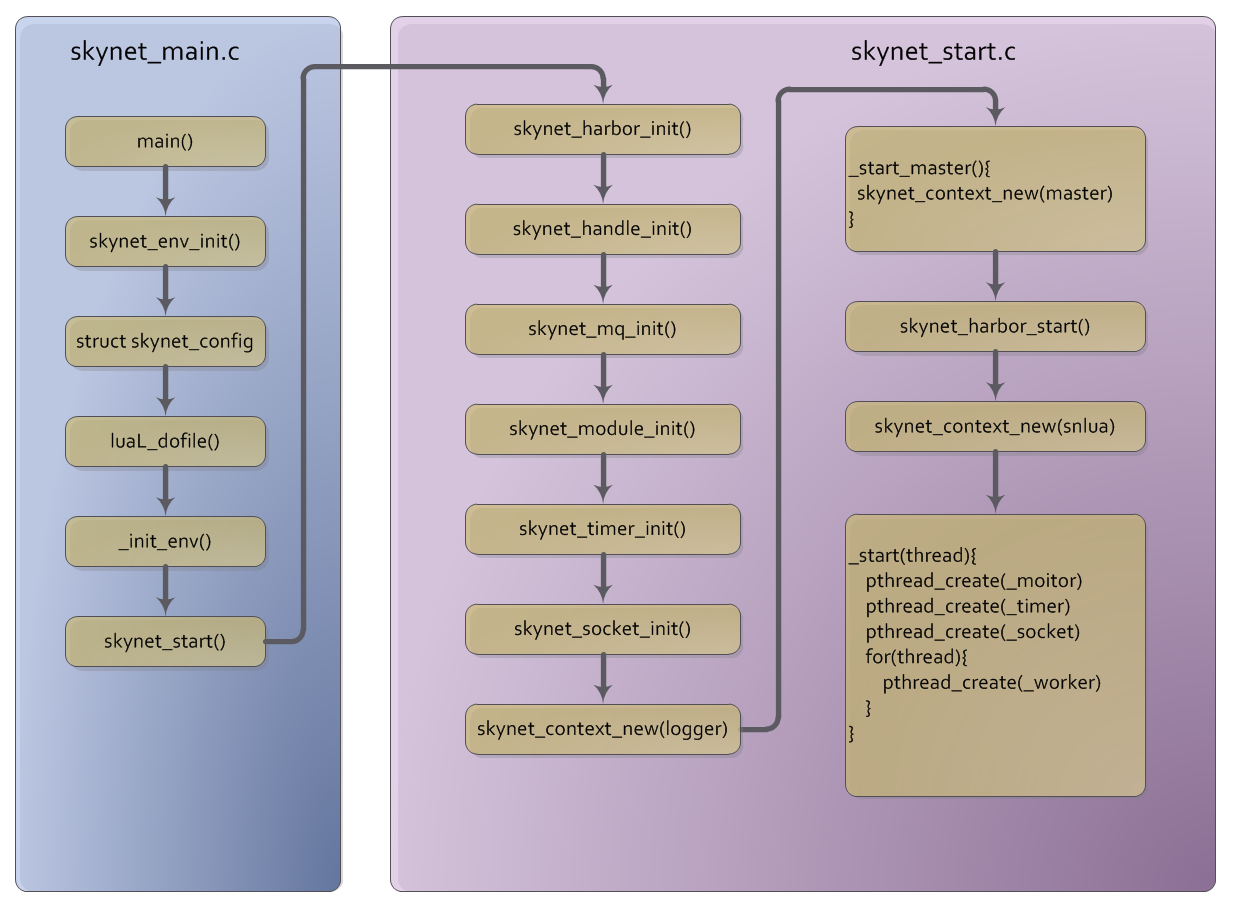


很简单的处理了SET和GET。

以上只是大概浏览了一遍Skynet附带的例子，了解了一些Skynet提供给lua使用的接口，其他接口可以查看skynet.lua代码。

# 六、原理

## 1、启动流程

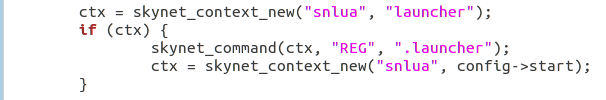


启动流程主要集中在两个文件中：

1是skynet-src/skynet\_main.c 这个是main()函数所在，主要就是设置一下lua的环境、默认的配置、打开config配置文件，并修改默认配置。最后调用skynet\_start()函数，这个函数在skynet\_start.c文件中。

2是skynet-src/skynet\_start.c这个文件主要是初始化Skynet的各个模块，包括harbor节点、handle服务ID、mq消息队列、module加载动态链接库、timer时钟、socket套接字以及加载一些服务logger日志服务、master主服务、harbor节点服务、snlua 加载lua模块的服务；以及最后启动几种线程包括\_moitor、\_timer、\_socket和根据线程数启动n个工作线程。

这里需要注意的是Skynet将通过snlua服务加载第一个用户的服务：



即：ctx = skynet\_context\_new("snlua", config->start); 这行，意思就是使用snlua加载config->start这个服务。而config->start指向配置文件config的start = "main"这行。

## 2、调用服务

这个主要在C语言中实现，代码在skynet-src/skynet\_server.c的skynet\_context\_new()函数中，这个函数主要就是实例化服务动态链接库中的”\_create()” 和 ”\_init()”，以及给服务创建一个私有的消息队列。并填充到struct skynet\_context结构中。这个结构很重要。

skynet\_send()，发送消息到队列中，等待调用服务的回调函数。

# 七、服务

## 1、用C语言写一个服务

用C语言编写一个服务，最简单最完整的就是service-src/server\_logger.c这个服务。

里面包含了：

1、logger\_create() 创建服务

2、logger\_init() 初始化服务

3、logger\_release() 释放服务

4、\_logger() 回调函数，这个名字有skynet\_callback(\_logger);决定。

一个服务中的这四个函数就是Skynet 服务动态链接库的API。这里的logger\_create()，其中logger表示的是logger.so的名字，Skynet的module会提取logger.so的名字作并加上”\_create”来识别服务中的函数地址。函数的执行顺序是先执行”\_create()”再执行”\_init()”。而”\_release”由skynet\_context\_release()调用来释放。而回调函数这是其他服务调用这个服务时会去调用它进行处理。服务的主要任务实现就在回调函数中处理。

## 2、用Lua语言写一个服务

用Lua语言写一个服务，最简单的例子在test/testsocket.lua这个服务，它主要实现了回射(echo)功能，客户端发送什么数据给服务端，服务端就像一面镜子把同样的数据反射回去给客户端。

skynet.start() 启动一个服务。