ETCD 学习笔记

王永刚

2019 年 12 月 20 日

目录

第一章　etcd 编译安装

1.1　golang 环境

当然在安装 etcd 前要先安装 go。并且设置好 GOPATH。可以用 apt,yum 或者源码安装，下载二进制安装等方式。

```
root@dockervm:~# go env
GOARCH="amd64"
GOBIN=""
GOCACHE="/root/.cache/go-build"
GOEXE=""
GOHOSTARCH="amd64"
GOHOSTOS="linux"
GOOS="linux"
GOPATH="/root/go"
GORACE=""
GOROOT="/usr/lib/go-1.10"
GOTMPDIR=""
GOTOOLDIR="/usr/lib/go-1.10/pkg/tool/linux_amd64"
GCCGO="gccgo"
CC="gcc"
CXX="g++"
CGO_ENABLED="1"
CGO_CFLAGS="-g -O2"
CGO_CPPFLAGS=""
CGO_CXXFLAGS="-g -O2"
CGO_FFLAGS="-g -O2"
CGO_LDFLAGS="-g -O2"
PKG_CONFIG="pkg-config"
GOGCCFLAGS="-fPIC -m64 -pthread -fmessage-length=0 -fdebug-prefix-map=/tmp/go-build406463150=/tmp/go-build -gno-record-gcc-swit
ches"
```

图 1.1: golang 环境变量

1.2　etcd 编译安装

```
#etcd 编译
mkdir -p  $GOPATH/src/go.etcd.io/
cd $GOPATH/src/go.etcd.io/
git clone https://github.com/etcd-io/etcd.git
./build
./bin/etcd
```

1.3　etcd 编译文件 build 分析

etcd 的 build 文件如图3.3所示。golang 的编译非常简洁快速。直接编译出了 etcd 和 etcdctl 两个可执行文件。

1.4　etcd 版本

本书开写时的 etcd 的最新版本。如果看代码，要用与本书一致比较好。

```
root@dockervm:~/go/src/go.etcd.io/etcd# ./bin/etcd
[WARNING] Deprecated '--logger=capnslog' flag is set; use '--logger=zap' flag instead
2019-12-12 12:56:57.536110 I | etcdmain: etcd Version: 3.5.0-pre
2019-12-12 12:56:57.536652 I | etcdmain: Git SHA: 378b05b8d
2019-12-12 12:56:57.536929 I | etcdmain: Go Version: go1.10.4
2019-12-12 12:56:57.537314 I | etcdmain: Go OS/Arch: linux/amd64
2019-12-12 12:56:57.537647 I | etcdmain: setting maximum number of CPUs to 1, total number of available CPUs is 1
2019-12-12 12:56:57.538025 W | etcdmain: no data-dir provided, using default data-dir ./default.etcd
[WARNING] Deprecated '--logger=capnslog' flag is set; use '--logger=zap' flag instead
2019-12-12 12:56:57.540593 I | embed: name = default
2019-12-12 12:56:57.541455 I | embed: data dir = default.etcd
2019-12-12 12:56:57.542031 I | embed: member dir = default.etcd/member
2019-12-12 12:56:57.542372 I | embed: heartbeat = 100ms
2019-12-12 12:56:57.542686 I | embed: election = 1000ms
2019-12-12 12:56:57.542978 I | embed: snapshot count = 100000
2019-12-12 12:56:57.543339 I | embed: advertise client URLs = http://localhost:2379
2019-12-12 12:56:57.861111 I | etcdserver: starting member 8e9e05c52164694d in cluster cdf818194e3a8c32
raft2019/12/12 12:56:57 INFO: 8e9e05c52164694d switched to configuration voters=()
raft2019/12/12 12:56:57 INFO: 8e9e05c52164694d became follower at term 0
raft2019/12/12 12:56:57 INFO: newRaft 8e9e05c52164694d [peers: [], term: 0, commit: 0, applied: 0, lastindex: 0, lastterm: 0]
raft2019/12/12 12:56:57 INFO: 8e9e05c52164694d became follower at term 1
raft2019/12/12 12:56:57 INFO: 8e9e05c52164694d switched to configuration voters=(10276657743932975437)
2019-12-12 12:56:58.012523 W | auth: simple token is not cryptographically signed
2019-12-12 12:56:58.128437 I | etcdserver: starting server... [version: 3.5.0-pre, cluster version: to_be_decided]
2019-12-12 12:56:58.134720 I | etcdserver: 8e9e05c52164694d as single-node; fast-forwarding 9 ticks (election ticks 10)
2019-12-12 12:56:58.135431 I | embed: listening for peers on 127.0.0.1:2380
raft2019/12/12 12:56:58 INFO: 8e9e05c52164694d switched to configuration voters=(10276657743932975437)
2019-12-12 12:56:58.136818 I | etcdserver/membership: added member 8e9e05c52164694d [http://localhost:2380] to cluster cdf81819
4e3a8c32
raft2019/12/12 12:56:58 INFO: 8e9e05c52164694d is starting a new election at term 1
raft2019/12/12 12:56:58 INFO: 8e9e05c52164694d became candidate at term 2
raft2019/12/12 12:56:58 INFO: 8e9e05c52164694d received MsgVoteResp from 8e9e05c52164694d at term 2
raft2019/12/12 12:56:58 INFO: 8e9e05c52164694d became leader at term 2
raft2019/12/12 12:56:58 INFO: raft.node: 8e9e05c52164694d elected leader 8e9e05c52164694d at term 2
2019-12-12 12:56:58.263397 I | etcdserver: setting up the initial cluster version to 3.5
2019-12-12 12:56:58.263744 I | etcdserver: published {Name:default ClientURLs:[http://localhost:2379]} to cluster cdf818194e3a8
c32
2019-12-12 12:56:58.264386 I | embed: ready to serve client requests
2019-12-12 12:56:58.265136 N | embed: serving insecure client requests on 127.0.0.1:2379, this is strongly discouraged!
2019-12-12 12:56:58.295344 N | etcdserver/membership: set the initial cluster version to 3.5
2019-12-12 12:56:58.295855 I | etcdserver/api: enabled capabilities for version 3.5
```

图 1.2: etcd 运行

```
49  etcd_build() {                          文件: go.etcd.io/etcd/build
50      out="bin"
51      if [[ -n "${BINDIR}" ]]; then out="${BINDIR}"; fi
52      toggle_failpoints_default
53
54      # Static compilation is useful when etcd is run in a container. $GO_BUILD_FLAGS is OK
55      # shellcheck disable=SC2086
56      CGO_ENABLED=0 go build $GO_BUILD_FLAGS \        build etcd: etcd/main.go -> etcd/etcdmain/etcdmain.go
57          -installsuffix cgo \
58          -ldflags "$GO_LDFLAGS" \
59          -o "${out}/etcd" ${REPO_PATH} || return
60      # shellcheck disable=SC2086
61      CGO_ENABLED=0 go build $GO_BUILD_FLAGS \
62          -installsuffix cgo \
63          -ldflags "$GO_LDFLAGS" \
64          -o "${out}/etcdctl" ${REPO_PATH}/etcdctl || return
65  }
```

图 1.3: etcd build

```
root@dockervm:~/go/src/go.etcd.io/etcd# ./bin/etcd --version
etcd Version: 3.5.0-pre
Git SHA: 378b05b8d
Go Version: go1.10.4
Go OS/Arch: linux/amd64
root@dockervm:~/go/src/go.etcd.io/etcd#
```

图 1.4: etcd 版本

第二章　etcd server 分析

2.1　etcd 启动调用关系

etcd 服务器启动，会启动很模块，对外主要是处理客户端请求，对内有 etcd 多个服务器进程之间通信，也有 etcd 服务器内部的功能，比如 KV 存储，WAL 日志等。源码分析第一步，搞清楚 etcd 的启动初始化步骤。见图2.1和图2.2。

```
(dlv) bt
0  0x0000000000b7db8b in go.etcd.io/etcd/etcdserver.(*EtcdServer).start
   at ./etcdserver/server.go:746
1  0x0000000000b7d8af in go.etcd.io/etcd/etcdserver.(*EtcdServer).Start
   at ./etcdserver/server.go:733
2  0x0000000000d307b9 in go.etcd.io/etcd/embed.StartEtcd
   at ./embed/etcd.go:228
3  0x0000000000d86ff0 in go.etcd.io/etcd/etcdmain.startEtcd
   at ./etcdmain/etcd.go:302
4  0x0000000000d85b6e in go.etcd.io/etcd/etcdmain.startEtcdOrProxyV2
   at ./etcdmain/etcd.go:144
5  0x0000000000d8fcff in go.etcd.io/etcd/etcdmain.Main
   at ./etcdmain/main.go:46
6  0x0000000000d94440 in main.main
   at ./main.go:28
7  0x0000000000042c5e2 in runtime.main
   at /usr/lib/go-1.10/src/runtime/proc.go:198
8  0x0000000000459e51 in runtime.goexit
   at /usr/lib/go-1.10/src/runtime/asm_amd64.s:2361
```

图 2.1: etcd 的 dlv 调用栈

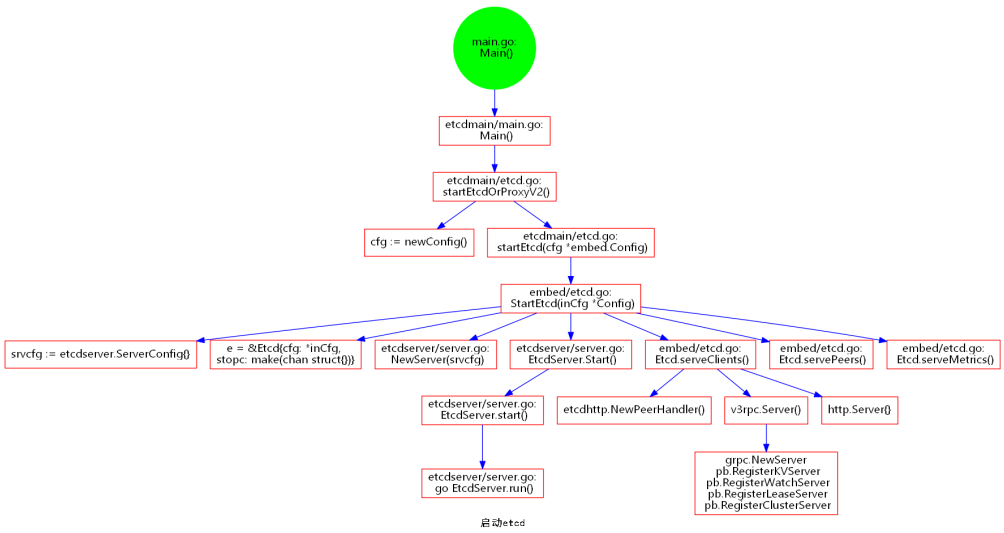

图 2.2: etcd 的启动初始化

第三章　etcd 编译安装

3.1　golang 环境

当然在安装 etcd 前要先安装 go。并且设置好 GOPATH。可以用 apt,yum 或者源码安装，下载二进制安装等方式。

```
root@dockervm:~# go env
GOARCH="amd64"
GOBIN=""
GOCACHE="/root/.cache/go-build"
GOEXE=""
GOHOSTARCH="amd64"
GOHOSTOS="linux"
GOOS="linux"
GOPATH="/root/go"
GORACE=""
GOROOT="/usr/lib/go-1.10"
GOTMPDIR=""
GOTOOLDIR="/usr/lib/go-1.10/pkg/tool/linux_amd64"
GCCGO="gccgo"
CC="gcc"
CXX="g++"
CGO_ENABLED="1"
CGO_CFLAGS="-g -O2"
CGO_CPPFLAGS=""
CGO_CXXFLAGS="-g -O2"
CGO_FFLAGS="-g -O2"
CGO_LDFLAGS="-g -O2"
PKG_CONFIG="pkg-config"
GOGCCFLAGS="-fPIC -m64 -pthread -fmessage-length=0 -fdebug-prefix-map=/tmp/go-build406463150=/tmp/go-build -gno-record-gcc-swit
ches"
```

图 3.1: golang 环境变量

3.2　etcd 编译安装

```
#etcd 编译
mkdir -p  $GOPATH/src/go.etcd.io/
cd $GOPATH/src/go.etcd.io/
git clone https://github.com/etcd-io/etcd.git
./build
./bin/etcd
```

3.3　etcd 编译文件 build 分析

etcd 的 build 文件如图3.3所示。golang 的编译非常简洁快速。直接编译出了 etcd 和 etcdctl 两个可执行文件。

3.4　etcd 版本

本书开写时的 etcd 的最新版本。如果看代码，要用与本书一致比较好。

Adobe Photoshop，简称“PS”，是由 Adobe Systems 开发和发行的图像处理软

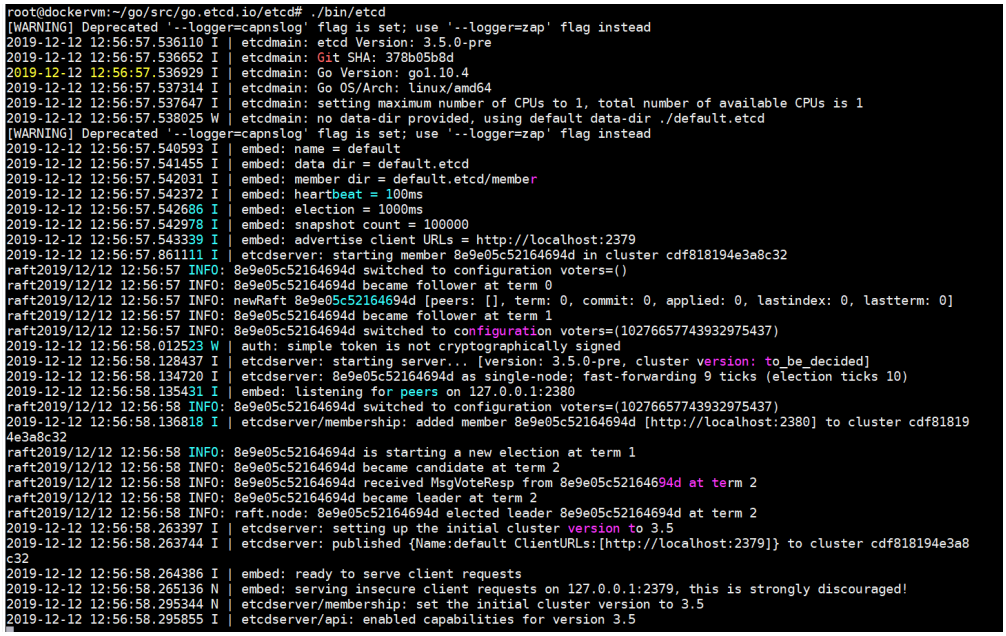

图 3.2: etcd 运行

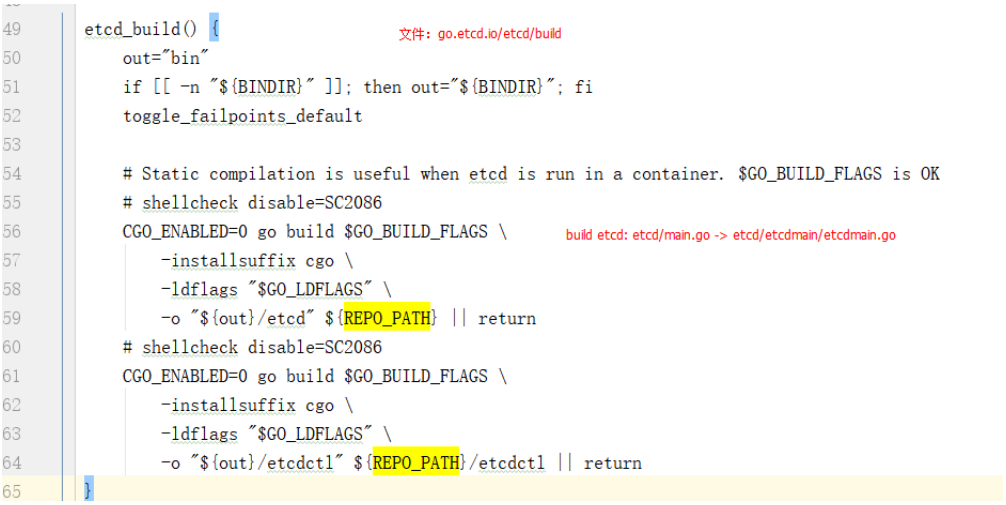

图 3.3: etcd build

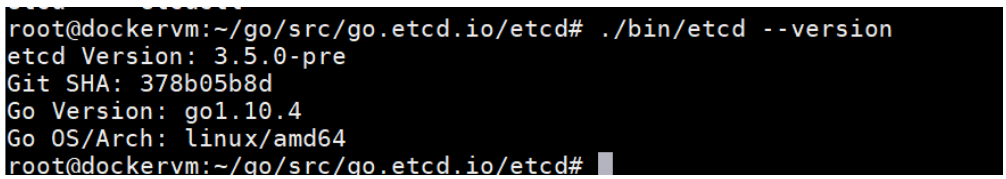

图 3.4: etcd 版本

件。Photoshop 主要处理以像素所构成的数字图像。使用其众多的编修与绘图工具，可以有效地进行图片编辑工作。PS 有很多功能，在图像、图形、文字、视频、出版等各方面都有涉及。2003 年，Adobe Photoshop 8 被更名为 Adobe Photoshop CS。2013 年 7 月，Adobe 公司推出了新版本的 Photoshop CC，自此，Photoshop CS6 作为 Adobe CS 系列的最后一个版本被新的 CC 系列取代。截止 2016 年 12

月 Adobe PhotoshopCC2017 所有数据类型见表3.1。至于详情可以参考 [1] 和 [2]。Adobe Photoshop，简称“PS”，是由 Adobe Systems 开发和发行的图像处理软

表 3.1: OpenFLow 包格式

序号	类型	描述
1	Packet-In	发送到控制器
2	Packet-Out	发到交换机
3	Flow-Mod	修改流表项，增删改查, 控制器发送到交换机的

件。Photoshop 主要处理以像素所构成的数字图像。使用其众多的编修与绘图工具，可以有效地进行图片编辑工作。PS 有很多功能，在图像、图形、文字、视频、出版等各方面都有涉及。2003 年，Adobe Photoshop 8 被更名为 Adobe Photoshop CS。2013 年 7 月，Adobe 公司推出了新版本的 Photoshop CC，自此，Photoshop CS6 作为 Adobe CS 系列的最后一个版本被新的 CC 系列取代。截止 2016 年 12 月 Adobe PhotoshopCC2017 为市场最新版本。跳转到第三章。这里是百度后面还有字这是楷体吗？

Adobe Photoshop，简称“PS”，是由 Adobe Systems 开发和发行的图像处理软件。Photoshop 主要处理以像素所构成的数字图像。使用其众多的编修与绘图工具，可以有效地进行图片编辑工作。PS 有很多功能，在图像、图形、文字、视频、出版等各方面都有涉及。2003 年，Adobe Photoshop 8 被更名为 Adobe Photoshop CS。2013 年 7 月，Adobe 公司推出了新版本的 Photoshop CC，自此，Photoshop CS6 作为 Adobe CS 系列的最后一个版本被新的 CC 系列取代。截止 2016 年 12 月 Adobe PhotoshopCC2017 为市场最新版本。这是楷体吗？

$$f(x) = 3x^2 + 6(x-2) - 1 \tag{3-1}$$

$$g(x) = 4x^2 + 6(x-8) + 1 \tag{3-2}$$

$$E = mc^2 \tag{3-3}$$

Adobe Photoshop，简称“PS”，是由 Adobe Systems 开发和发行的图像处理软件。Photoshop 主要处理以像素所构成的数字图像。使用其众多的编修与绘图工具，可以有效地进行图片编辑工作。PS 有很多功能，在图像、图形、文字、视频、出版等各方面都有涉及。2003 年，Adobe Photoshop 8 被更名为 Adobe Photoshop CS。2013 年 7 月，Adobe 公司推出了新版本的 Photoshop CC，自此，Photoshop CS6 作为 Adobe CS 系列的最后一个版本被新的 CC 系列取代。截止 2016 年 12 月 Adobe PhotoshopCC2017 为市场最新版本。质能公式如公式(3-3)。这是楷体吗？

3.4.1　画布操作

1. 新建画布　Ctrl + N

2. 画布切换　F
3. 复位工作区　Alt -> W -> K -> R
4. 放大缩小　Alt + 鼠标滚轮 (Ctrl + +, Ctrl + -)
5. 缩放工具　Z(放大：鼠标点击画布，或按下 Alt 点击画布，可以放大缩小)
6. 移动画布　Space + 鼠标左键按下拖动
7. 切换画布　Ctrl + Tab
8. 显示网格　Ctrl + ’
9. 显示参考线　Ctrl + ;
10. 显示标尺　Ctrl + R

Adobe Photoshop，简称“PS”，是由 Adobe Systems 开发和发行的图像处理软件。Photoshop 主要处理以像素所构成的数字图像。使用其众多的编修与绘图工具，可以有效地进行图片编辑工作。PS 有很多功能，在图像、图形、文字、视频、出版等各方面都有涉及。2003 年，Adobe Photoshop 8 被更名为 Adobe Photoshop CS。2013 年 7 月，Adobe 公司推出了新版本的 Photoshop CC，自此，Photoshop CS6 作为 Adobe CS 系列的最后一个版本被新的 CC 系列取代。截止 2016 年 12 月 Adobe PhotoshopCC2017 为市场最新版本。这是楷体吗？

Adobe Photoshop，简称“PS”，是由 Adobe Systems 开发和发行的图像处理软件。Photoshop 主要处理以像素所构成的数字图像。使用其众多的编修与绘图工具，可以有效地进行图片编辑工作。PS 有很多功能，在图像、图形、文字、视频、出版等各方面都有涉及。2003 年，Adobe Photoshop 8 被更名为 Adobe Photoshop CS。2013 年 7 月，Adobe 公司推出了新版本的 Photoshop CC，自此，Photoshop CS6 作为 Adobe CS 系列的最后一个版本被新的 CC 系列取代。截止 2016 年 12 月 Adobe PhotoshopCC2017 为市场最新版本。 Adobe Photoshop，简称“PS”，是

图 3.5: 风景 1

图 3.6: 风景 2

由 Adobe Systems 开发和发行的图像处理软件。Photoshop 主要处理以像素所构成的数字图像。使用其众多的编修与绘图工具，可以有效地进行图片编辑工作。PS 有很多功能，在图像、图形、文字、视频、出版等各方面都有涉及。2003 年，Adobe Photoshop 8 被更名为 Adobe Photoshop CS。2013 年 7 月，Adobe 公司推出了新版本的 Photoshop CC，自此，Photoshop CS6 作为 Adobe CS 系列的最后一个版本被新的 CC 系列取代。截止 2016 年 12 月 Adobe PhotoshopCC2017 为市场最新版本。如图3.5所示，又如图3.6所示。这是楷体吗？

Adobe Photoshop，简称“PS”，是由 Adobe Systems 开发和发行的图像处理软件。Photoshop 主要处理以像素所构成的数字图像。使用其众多的编修与绘图工具，可以有效地进行图片编辑工作。PS 有很多功能，在图像、图形、文字、视频、出版等各方面都有涉及。2003 年，Adobe Photoshop 8 被更名为 Adobe Photoshop CS。2013 年 7 月，Adobe 公司推出了新版本的 Photoshop CC，自此，Photoshop CS6 作为 Adobe CS 系列的最后一个版本被新的 CC 系列取代。截止 2016 年 12 月 Adobe PhotoshopCC2017 为市场最新版本。这是楷体吗？

Adobe Photoshop，简称“PS”，是由 Adobe Systems 开发和发行的图像处理软件。Photoshop 主要处理以像素所构成的数字图像。使用其众多的编修与绘图工具，可以有效地进行图片编辑工作。PS 有很多功能，在图像、图形、文字、视频、出版等各方面都有涉及。2003 年，Adobe Photoshop 8 被更名为 Adobe Photoshop CS。2013 年 7 月，Adobe 公司推出了新版本的 Photoshop CC，自此，Photoshop CS6 作为 Adobe CS 系列的最后一个版本被新的 CC 系列取代。截止 2016 年 12 月 Adobe PhotoshopCC2017 为市场最新版本。这是楷体吗？

3.4.2　phtotshop 简介

参考文献

[1] 广西壮族自治区林业厅. 广西自然保护区 [M]. 北京: 中国林业出版社, 1993.

[2] International Federation of Library Association and Institutions. Names of persons: national usages for entry in catalogues[M]. 3rd ed. London: IFLA International Office for UBC, 1977.

[3] GANZHA V G, MAYR E W, VOROZHTSOV E V. Computer algebra in scientific computing: CASC 2000: proceedings of the Third Workshop on Computer Algebra in Scientific Computing, Samarkand, October 5-9, 2000[C]. Berlin: Springer, c2000.

附录

C 语言结构体用法

```c
#include <stdio.h>
#include <string.h>

struct Books
{
    char  title[50];
    char  author[50];
    char  subject[100];
    int   book_id;
};

int main( )
{
    struct Books Book1;        /* 声明 Book1, 类型为 Books */
    struct Books Book2;        /* 声明 Book2, 类型为 Books */

    /* Book1 详述 */
    strcpy( Book1.title, "C Programming");
    strcpy( Book1.author, "Nuha Ali");
    strcpy( Book1.subject, "C Programming Tutorial");
    Book1.book_id = 6495407;

    /* Book2 详述 */
    strcpy( Book2.title, "Telecom Billing");
    strcpy( Book2.author, "Zara Ali");
    strcpy( Book2.subject, "Telecom Billing Tutorial");
    Book2.book_id = 6495700;

    /* 输出 Book1 信息 */
    printf( "Book 1 title : %s\n", Book1.title);
    printf( "Book 1 author : %s\n", Book1.author);
    printf( "Book 1 subject : %s\n", Book1.subject);
    printf( "Book 1 book_id : %d\n", Book1.book_id);

    /* 输出 Book2 信息 */
    printf( "Book 2 title : %s\n", Book2.title);
    printf( "Book 2 author : %s\n", Book2.author);
    printf( "Book 2 subject : %s\n", Book2.subject);
    printf( "Book 2 book_id : %d\n", Book2.book_id);

    return 0;
}
```

示例 2 C# 代码

```
string title = "This is a Unicode  in the sky"
/*
Defined as π = lim_{n→∞} P_n/d where P is the perimeter
of an n-sided regular polygon circumscribing a
circle of diameter d.
*/
const double pi = 3.1415926535
```

```
#include  <stdio.h>

int main(){
    int  sum=0;
    int  num=1;
    int  sum2=0;
    int  num2=2;
    while(num<100){
        sum=sum+num;
        num=num+2;
    }
    printf("奇数和为:%d\n",sum);

    while(num2<=100){
        sum2=sum2+num2;
        num2=num2+2;
    }
    printf("偶数和为: %d\n",sum2);
}
```

```c
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <unistd.h>

#include <json.h>

#include "pubnub.h"
#include "pubnub-sync.h"

int
main(void)
{
        struct pubnub_sync *sync = pubnub_sync_init();
        struct pubnub *p = pubnub_init(
                        /* publish_key */ "demo",
                        /* subscribe_key */ "demo",
                        /* pubnub_callbacks */
                         ↪  &pubnub_sync_callbacks,
                        /* pubnub_callbacks data */ sync);
        json_object *msg;


        /* Publish */

        msg = json_object_new_object();
```

```
json_object_object_add(msg, "num",
 ↪  json_object_new_int(42));
json_object_object_add(msg, "str",
 ↪  json_object_new_string("\"Hello, world!\" she said."));

pubnub_publish(
                /* struct pubnub */ p,
                /* channel */ "my_channel",
                /* message */ msg,
                /* default timeout */ -1,
                /* callback; sync needs NULL! */ NULL,
                /* callback data */ NULL);

json_object_put(msg);

if (pubnub_sync_last_result(sync) != PNR_OK)
        return EXIT_FAILURE;
msg = pubnub_sync_last_response(sync);
printf("pubnub publish ok: %s\n",
 ↪  json_object_get_string(msg));
json_object_put(msg);


/* History */

pubnub_history(
                /* struct pubnub */ p,
```

```
                /* channel */ "my_channel",
                /* #messages */ 10,
                /* default timeout */ -1,
                /* callback; sync needs NULL! */ NULL,
                /* callback data */ NULL);
if (pubnub_sync_last_result(sync) != PNR_OK)
        return EXIT_FAILURE;
msg = pubnub_sync_last_response(sync);
printf("pubnub history ok: %s\n",
 ↪  json_object_get_string(msg));
json_object_put(msg);


/* Subscribe */

do {
        const char *channels[] = { "my_channel",
         ↪  "demo_channel" };
        pubnub_subscribe_multi(
                        /* struct pubnub */ p,
                        /* list of channels */ channels,
                        /* number of listed channels */ 2,
                        /* default timeout */ -1,
                        /* callback; sync needs NULL! */
                         ↪  NULL,
                        /* callback data */ NULL);
        if (pubnub_sync_last_result(sync) != PNR_OK)
```

```
                return EXIT_FAILURE;
            msg = pubnub_sync_last_response(sync);
            if (json_object_array_length(msg) == 0) {
                printf("pubnub subscribe ok, no news\n");
            } else {
                char **msg_channels =
                ↪ pubnub_sync_last_channels(sync);
                for (int i = 0; i <
                ↪ json_object_array_length(msg); i++) {
                    json_object *msg1 =
                    ↪ json_object_array_get_idx(msg,
                    ↪ i);
                    printf("pubnub subscribe [%s]:
                    ↪ %s\n", msg_channels[i],
                    ↪ json_object_get_string(msg1));
                }
            }
            json_object_put(msg);
            sleep(1);
        } while (1);


        pubnub_done(p);
        return EXIT_SUCCESS;
}
```

致谢

谢谢！

作者简介