目录

[1 Linux快捷键 2](#_Toc2713)

[1.1 文本编辑 2](#_Toc25712)

[1.2编译 2](#_Toc2946)

[1.3 github 2](#_Toc18617)

[1.3.1 Github 变化代码提交 3](#_Toc18749)

[2 优质代码 3](#_Toc28902)

[2.1 位设置 3](#_Toc18746)

[3 字节序 4](#_Toc12675)

[4 回调 5](#_Toc18248)

[5 强制类型转换 5](#_Toc29389)

[5.1 类型转换 5](#_Toc24497)

[5.2 寄存器操作 5](#_Toc26848)

# Linux快捷键

## 文本编辑

文本编辑可以使用命令gedit这是ubuntu自带的文本编辑工具，也可以使用vim，vim号称最强大的文本编辑工具。但是gedit在ubuntu下使用更加方便。

## 1.2编译

gcc main.c -o main

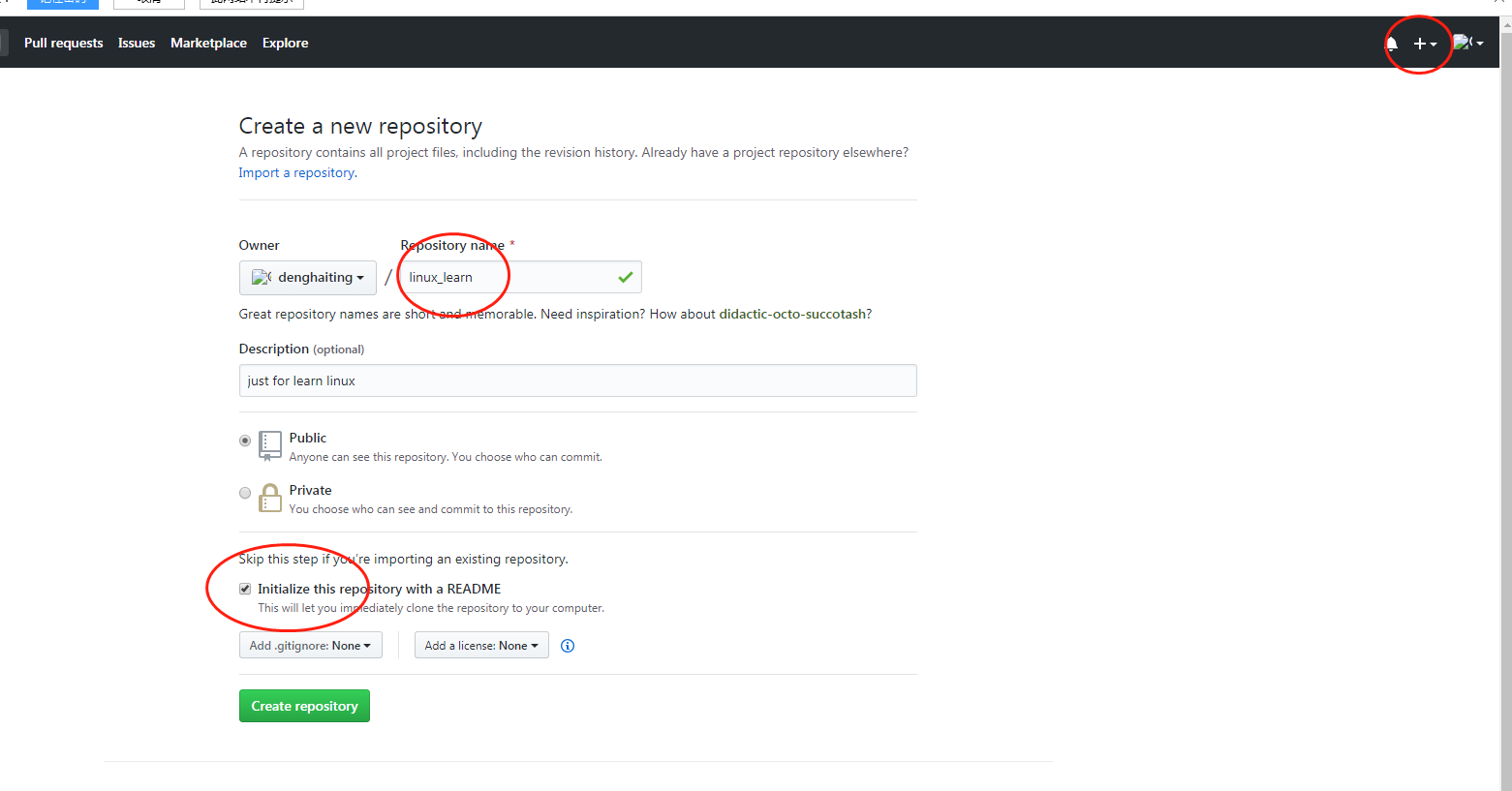
## 1.3 github

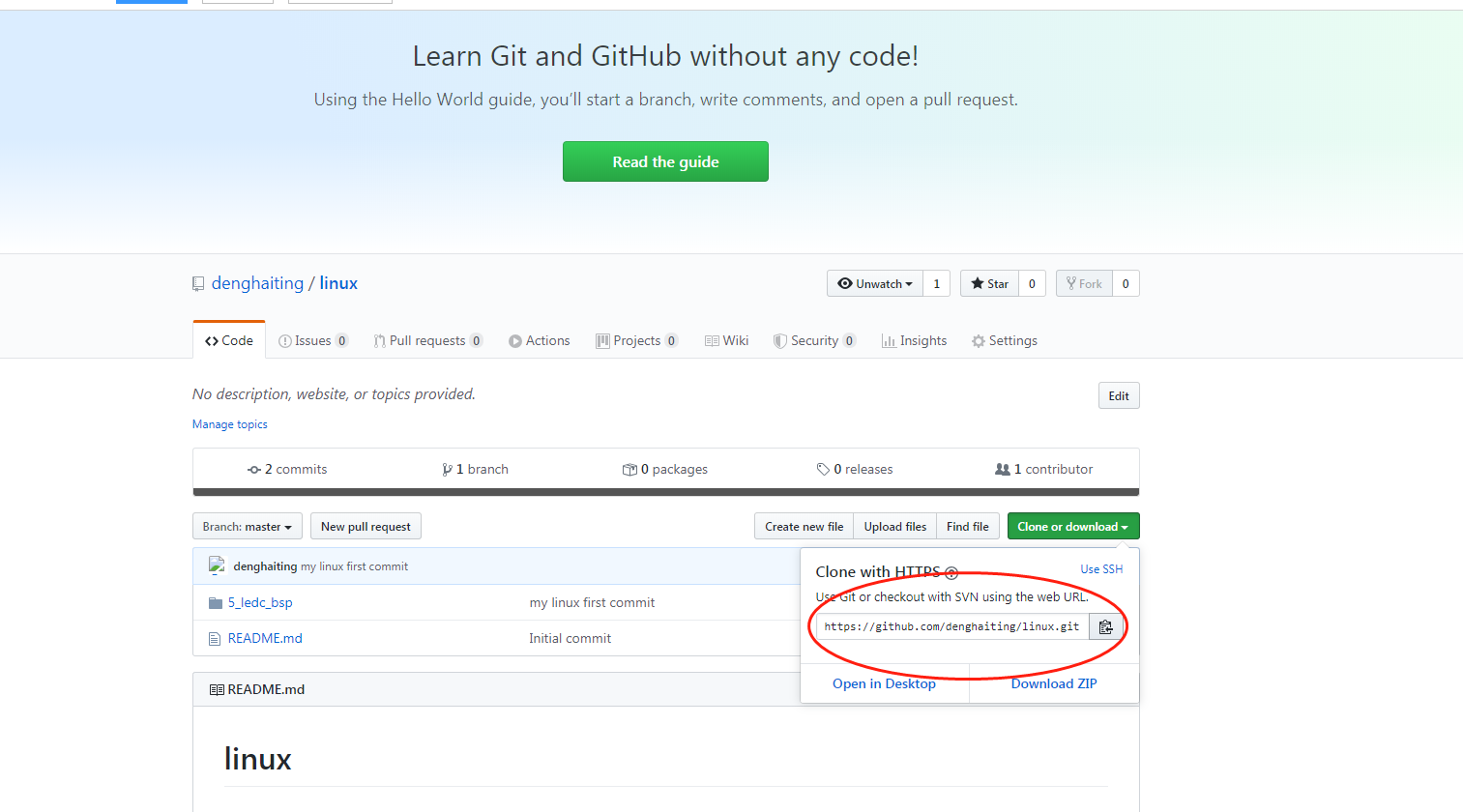
2.初次运行Git需要对Git环境进行配置，具体指令为：

$ git config --global user.name "用户名" $ git config --global user.email 自己的邮箱

首先在github网站上创建一个仓库，git账号：denghaiting,password:\*\*\*\*\*\*dhtt. 在主页上会有一个加号，点击加号 new repository创建一个新的仓库如新建一个叫linux的仓库，然后进入点击clone or download,然后点击复制按钮。然后在ubutu下输入

命令：git clone https://github.com/denghaiting/linux.git那么就在你ubuntu的本地目录中有了一个linux的文件夹，将自己写的代码放入该文件夹下，然后输入命令;git add 文件名或git add \*，然后输入命令：git commit -m "my linux first commit" 引号中的内容为提交备注。然后在使用git push 推送到远程服务器。之后要输入账号和密码。git pull从gitHub服务器更新代码到本地





### 1.3.1 Github 变化代码提交

当本地代码修改之后使用命令：git add -A然后再使用命令：git status 再次：git commit -m “注释”

再次git push

如果想要在仓库下添加一个文件夹（此文件夹必须非空）必须 git add 文件夹名 ，然后git commit -m “注释”再次 git push

这个网址是讲如何在linux上git代码：<https://blog.csdn.net/Sophiahaha/article/details/83041505>

### 1.3.2如何从 Github 上更新最新代码

如果之前已经从github上获取了最新的代码，现在想要更新最新的文件那么可以使用git checkout 文件名

如:git checkout doc git checkout rizhi.txt

# 优质代码

## 位设置

#define setbit(x,y) ((x) |= (1<<(y)))

#define clrbit(x,y) ((x) &= ~(1<<(y)))

#define setbit\_array(x,y) (setbit((x)[(y)/8],(y)%8))//设置数组的第几位为1

#define clrbit\_array(x,y) (clrbit((x)[(y)/8],(y)%8))//设置数组的第几位为0

例如：uint8\_t analog\_cfg\_buf[16]={0};setbit\_array(analog\_cfg\_buf,120);

# 字节序

#ifndef htons

#define htons(x) ((uint16\_t)( \

(((uint16\_t)(x) & (uint16\_t)0x00ffU) << 8) | \

(((uint16\_t)(x) & (uint16\_t)0xff00U) >> 8)))

#endif /\* htonl \*/

#ifndef ntohs

#define ntohs(x) ((uint16\_t)( \

(((uint16\_t)(x) & (uint16\_t)0x00ffU) << 8) | \

(((uint16\_t)(x) & (uint16\_t)0xff00U) >> 8)))

#endif /\* ntohl \*/

#ifndef htonl

#define htonl(x) ((uint32\_t)( \

(((uint32\_t)(x) & (uint32\_t)0x000000ffUL) << 24) | \

(((uint32\_t)(x) & (uint32\_t)0x0000ff00UL) << 8) | \

(((uint32\_t)(x) & (uint32\_t)0x00ff0000UL) >> 8) | \

(((uint32\_t)(x) & (uint32\_t)0xff000000UL) >> 24)))

#endif /\* htons \*/

#ifndef ntohl

#define ntohl(x) ((uint32\_t)( \

(((uint32\_t)(x) & (uint32\_t)0x000000ffUL) << 24) | \

(((uint32\_t)(x) & (uint32\_t)0x0000ff00UL) << 8) | \

(((uint32\_t)(x) & (uint32\_t)0x00ff0000UL) >> 8) | \

(((uint32\_t)(x) & (uint32\_t)0xff000000UL) >> 24)))

#endif /\* ntohs \*/

# 回调

看macbee协议库函数和门磁应用程序全局搜索exti\_cb即可查看irq\_callback\_t这个回调是怎么回事macbee/stm32f0/trunk/library

# 强制类型转换

## 类型转换

Uint8\_t group[4]; uint32\_t group\_start\_seq = 0;

typedef struct{

uint8\_t cmd;

uint8\_t result;

uint16\_t enc\_sel;

uint32\_t time\_UTC;

uint8\_t groups[4];

uint32\_t group\_start\_seq;

} gw\_auth\_result\_t;

typedef struct {

uint32\_t rmt\_ID;

} led\_config\_t;

led\_config\_t led\_config;

\*(uint32\_t\*)groups = \*(uint32\_t\*)((gw\_auth\_result\_t\*)rx\_buf)->groups;

group\_start\_seq = ((gw\_auth\_result\_t\*)rx\_buf)->group\_start\_seq;

((uint8\_t \*)(&led\_config.rmt\_ID))[2] = 0;

\*(unsigned\*)0xE000ED88 |= ((3UL << 10\*2)|(3UL << 11\*2));

\*( volatile Uint32\* )( 0x01C40000+ 0x0C )  &= ~0x00600000;

## 寄存器操作

#define MMIO32(addr) (\*(volatile uint32\_t \*)(addr))

#define MACBEE\_RF\_BASE 0x40020000

#define CFG\_TOP MMIO32(MACBEE\_RF\_BASE + 0X00)