# 生成patch并提交

## 准备

开始之前，需要具备了如下的工具：

* 文本编辑器
* Email 客户端
* 版本控制系统（例如：git）
* Github账号

## 下载内核代码仓库

### 2.1 配置email邮箱

#### 2.1.1 设置git身份和邮箱

git config --global user.name "yourname"

git config --global user.email“your@email.com"

注：yourname是你要设置的名字，your@email是你要设置的邮箱。



#### 清空

删除.ssh文件夹（直接搜索该文件夹）下的known\_hosts(手动删除即可，不需要git）

#### git输入命令

1）$ ssh-keygen -t rsa -C "your@email.com"（请填你设置的邮箱地址）

2）接着出现：

Generating public/private rsa key pair.

Enter file in which to save the key (/Users/your\_user\_directory/.ssh/id\_rsa):

3）请直接按下回车

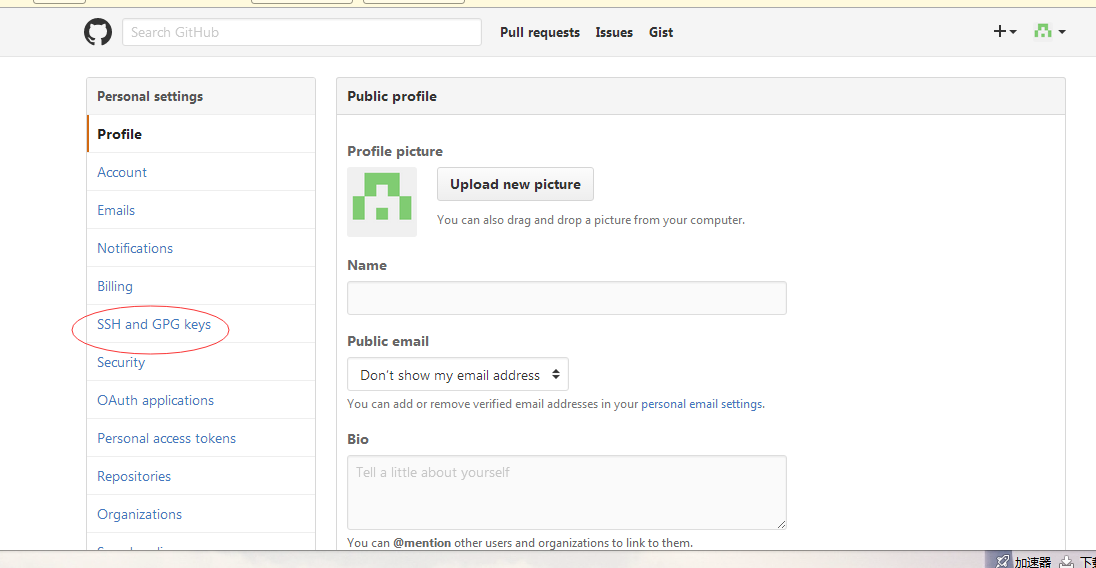
4）然后系统会自动在.ssh文件夹下生成两个文件，id\_rsa和id\_rsa.pub，用记事本打开id\_rsa.pub

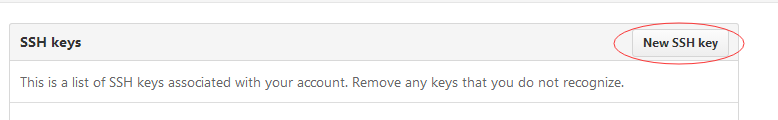
5）将全部的内容复制

#### 2.1.4 设置github

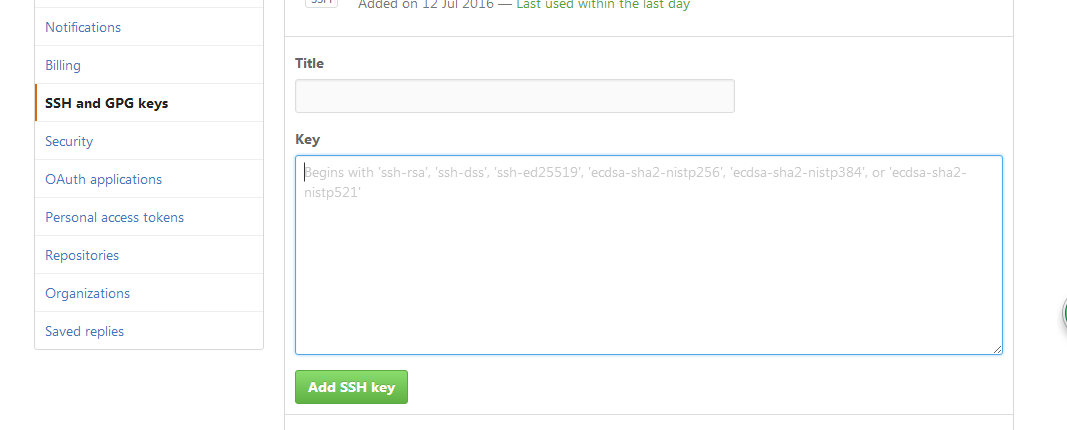
打开https://github.com/，登陆你的账户，进入设置；

进入ssh设置





在key中将刚刚复制的粘贴进去



点击add ssh key

#### 2.1.5 输入git命令

在git中输入命令：

ssh -T [git@github.com](mailto:git@github.com)

然后会跳出一堆话。

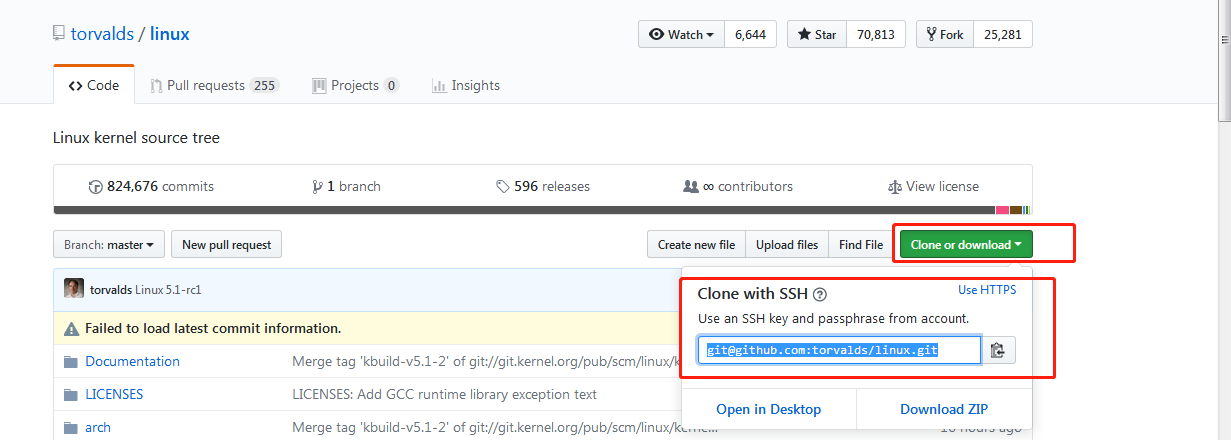
输入命令：yes

回车

### 下载内核仓库

#### 2.2.1 下载地址

进入<https://github.com/>，搜索linux，找到[torvalds](https://github.com/torvalds)/[linux](https://github.com/torvalds/linux)



#### 下载

* 1. 使用命令：

git clone [git@github.com:torvalds/linux.git](mailto:git@github.com:torvalds/linux.git)

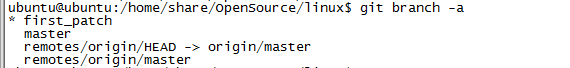
下载需要很长时间；

* 1. 查看所有分支列表

**git branch –a**

* 1. 创建并切换到其他分支

git checkout -b first\_patch



* 1. 查看当前仓库中文件的状态

git status



* 1. 切换到主分支

git checkout master

* 1. 切换到其他分支

git checkout first\_patch

* 1. **更新远程跟踪分支**

git fetch origin

git rebase origin/master

## 修改，编译

1. 在自己拉出的分支中，对文件进行修改
2. 使用 make 命令重新编译，确保你的变更没有错误。
3. 安装
4. 使用新版本内核进行启动

## 提交并创建patch

### 提交

git add drivers/net/vxlan.c

git add <file>

git commit -s –v

注意 git commit 命令会自动打开编辑器让你编辑 Commit 信息，-s 参数可以自动在你的 Commit 信息下加上一行Signed-off-by: My Name <my\_email@gmail.com> ，-v 参数会在你的 Commit 信息下方显示出你做的修改，确保你能再三检查自己的改动，这一个参数不是必须的，但是推荐这么做。

注意，Commit 信息的格式有严格限制，我就不废话了，直接上模板。

mm: fix some error

Why I do these changes and how I do it.

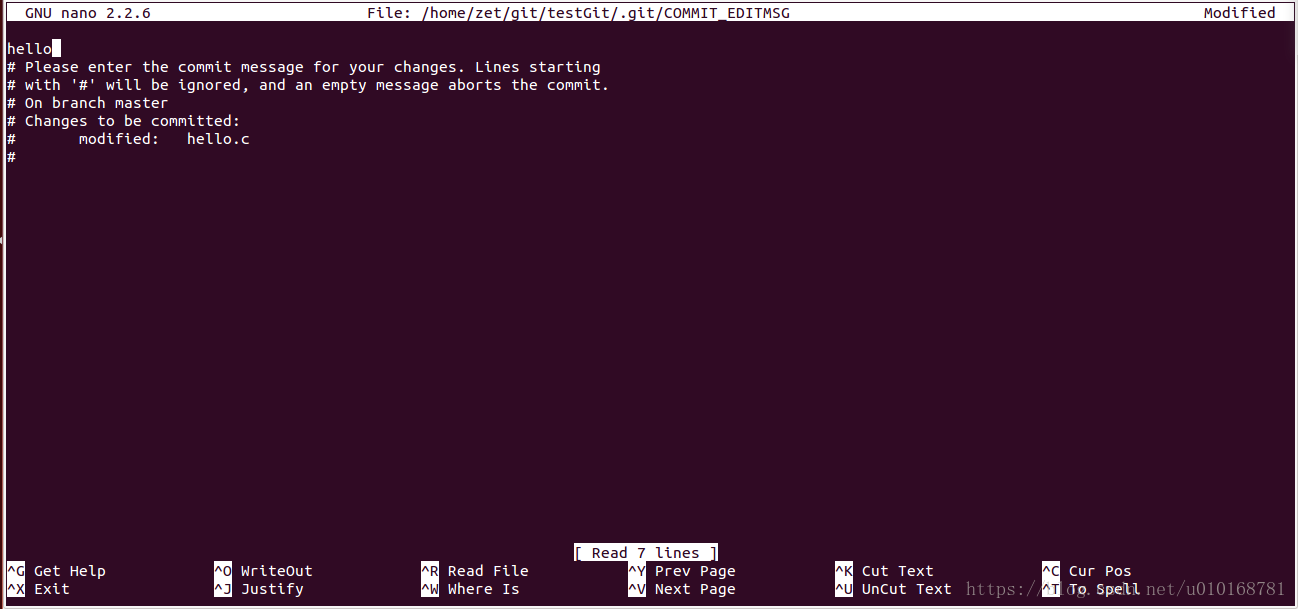
Signed-off-by: My Name <my\_email@gmail.com>

1. 第一部分是 short description，以子系统名打头，比如 mm，注意分号后面加个空格，不知道子系统名的可以看看你修改的这个文件的修改历史，看看之前的开发者是怎么写的。这一部分需要使用一句简短的话描述你所做的修改，要让维护者一眼就看出这个 Patch 大概干了什么事。
2. 第二部分是 the body of your patch，这一部分要详细的解释你为何要做这个修改，以及怎么做的，注意时态用现在时，语态用主动形式。
3. 第三部分是之前的 -s 参数自动加上的，不用管。
4. 必须要注意的是，这三部分之间都要有一个空行隔开。

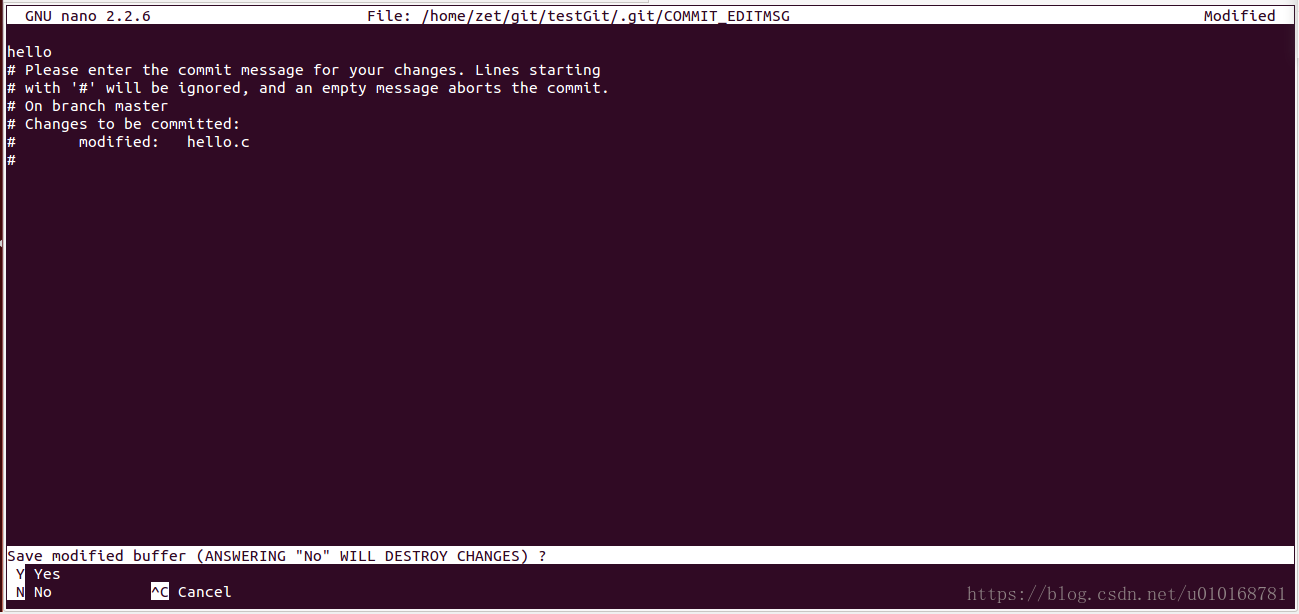
如果 commit 之后还想修改 Commit 信息的话需要使用命令 git commit --amend -v

编辑完后保存退出（使用 git commit 命令后，进入的是nano文本编辑器）

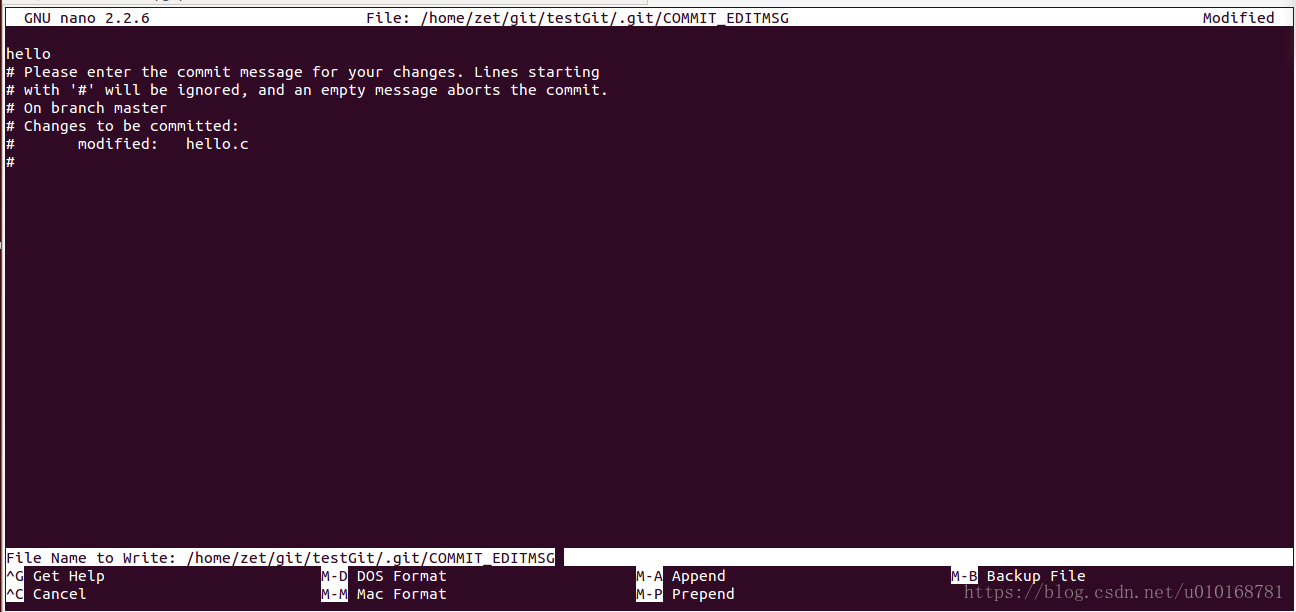
### nano文本编辑器的保存退出



填写修改记录，按下Ctrl+x



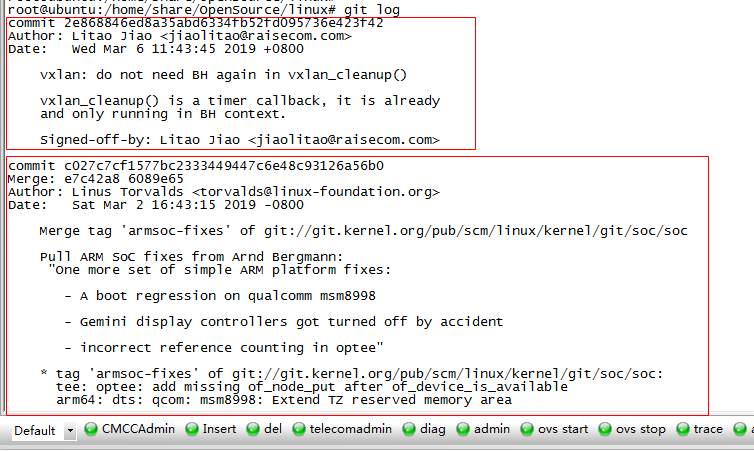
输入Y保存，进入如下界面，然后回车退出



git默认的文本编辑器是nano，执行下面的命令将git的文本编辑器改为我们熟悉的vim  
git config --global core.editor vim

### 查看提交的log信息

git log



### 生成patch

git format-patch -o /tmp/ HEAD^

### 检查 Patch 格式

运行以下命令检查你的 Patch 格式有没有问题，要做到 0 errors, 0 warnings。

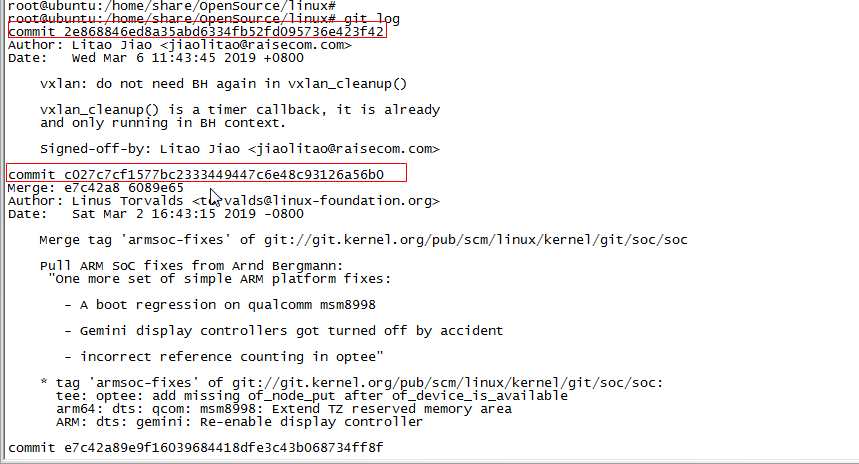
./scripts/checkpatch.pl 0001-vxlan-do-not-need-BH-again-in-vxlan\_cleanup.patch

|  |
| --- |
| total: 0 errors, 0 warnings, 0 checks, 16 lines checked  0001-vxlan-do-not-need-BH-again-in-vxlan\_cleanup.patch has no obvious style problems and is ready for submission. |

### 回退提交

如果提交后，发现提交有问题，可以进行回退

先使用git log获取以前提交的信息



如果发现上次提交有误，则使用：

git reset --hard c027c7cf1577bc2333449447c6e48c93126a56b0

回退到上一个版本中

## 发送patch

既然 Patch 已经生成完毕，那么是时候发送给上游维护者了。

### 寻找上游维护者

运行以下命令找出你应该把 Patch 发给谁。

./scripts/get\_maintainer.pl -f include/linux/gfp.h

注意，include/linux/gfp.h 这个文件名改成你所修改的文件

该命令输出如下：

Andrew Morton <akpm@linux-foundation.org> (commit\_signer:7/8=88%)

Michal Hocko <mhocko@[SUSE](http://www.linuxidc.com/topicnews.aspx?tid=3).com> (commit\_signer:3/8=38%,authored:1/8=12%,added\_lines:3/21=14%,removed\_lines:10/33=30%)

Vlastimil Babka <vbabka@suse.cz> (commit\_signer:3/8=38%,authored:1/8=12%,added\_lines:8/21=38%,removed\_lines:6/33=18%)

Alexander Duyck <alexander.h.duyck@intel.com> (commit\_signer:3/8=38%,authored:3/8=38%,added\_lines:6/21=29%,removed\_lines:5/33=15%)

Mel Gorman <mgorman@suse.de> (commit\_signer:2/8=25%)

Vladimir Davydov <vdavydov.dev@gmail.com> (authored:1/8=12%,removed\_lines:9/33=27%)

My Name <my\_email@gmail.com> (authored:1/8=12%,added\_lines:2/21=10%,removed\_lines:2/33=6%)

linux-mm@kvack.org (open list:MEMORY MANAGEMENT)

linux-kernel@vger.kernel.org (open list)

### 测试发送

最好在发送给上游维护者之前先拿自己邮箱做个测试，小心点总没坏处。

git send-email --to my\_qq@qq.com 0001-mm-fix-some-error.patch

一切正常的话，你应该可以收到邮件了，检查下格式什么的是否和你预想的一样。

### 正式发送

git send-email --to akpm@linux-foundation.org \

--cc alexander.h.duyck@intel.com \

--cc mhocko@suse.com --cc vbabka@suse.cz \

--cc mgorman@suse.de \

--cc l.stach@pengutronix.de \

--cc vdavydov.dev@gmail.com \

--cc linux-mm@kvack.org \

--cc linux-kernel@vger.kernel.org \

--cc my\_email@gmail.com \

0001-mm-fix-some-error.patch

之后你的 Patch 就发送给上游维护者并抄送到对应的邮件列表了。

如果 Patch 并入上游分支的话会给你发邮件通知；

## 参考文献

https://mp.weixin.qq.com/s/34Gna2jHGZ82b1wqkXrJqw

https://linux.cn/article-10109-1.html

https://kernelnewbies.org/FirstKernelPatch

https://www.linuxidc.com/Linux/2017-04/142492.htm

https://blog.csdn.net/mirage1993/article/details/53957205