软件与微电子学院
集成电路与智能系统系

《嵌入式与物联网操作系统》

代码分析报告

小　组：lung good team

成　员：杭锦泉、滕飞、朱君楷

日　期：2020.3.12

一、实验目的

1、在建立交叉编译开发平台之前，首先需要建立主机(虚拟机/PC，或双系统）开发环境；
2、学会使用本地 gcc 编译应用程序；
3、学会使用 Makefile 管理应用程序；
4、学会通过 autotools 生成 Makefile，学会常用的 make 操作；
5、学会通过 git/github 管理团队软件和工作文件。

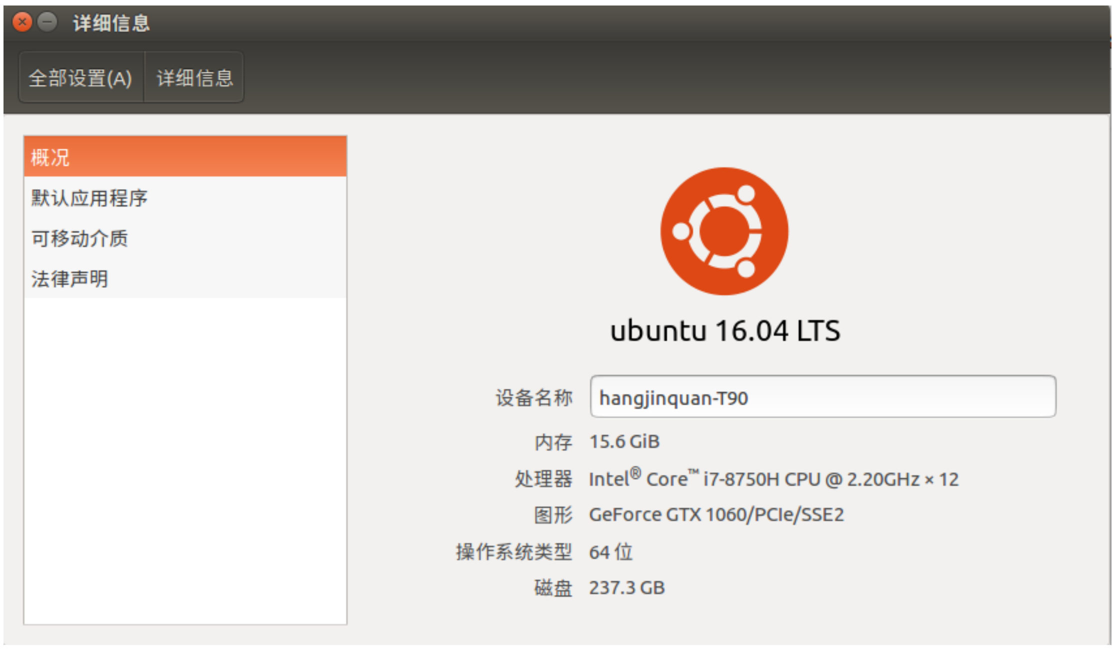
二、实验内容

1、安装主机(虚拟机/PC)Linux 开发环境，Fedora，Ubuntu ，Debian 均可；
2、编写 c 应用程序，通过本地 gcc 编译应用程序，如果是基于 x86 的主机，gcc 输出的执行文件运行的是 x86 指令集；
3、编写 Makefile 管理应用程序，为更好体现 Makefile 的作用，需编写多个 c 程序，给出所创建的 Makefile 的内容；
4、通过 autotools 生成 Makefile，完成常用的 make 操作(make, make install, make uninstall, make dist)；
5、创建小组 git 仓库，github 账号，用来存储小组工作文件以及小组报告；学习如何构建 github 文件，如何上传和下载 github 文件等。

三、实验过程与结果

1、安装主机(虚拟机/PC)Linux 开发环境：

在 VMareWorkstation 上安装 ubuntu 16.04LTS 的虚拟机，环境截图如下：

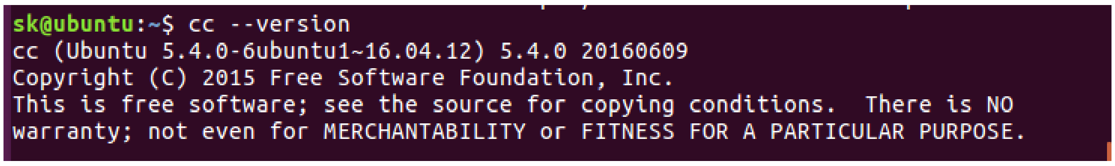


设备名称 hangjinquan-T90
内存 15.6 GiB
处理器 Intel® Core™ i7-8750H CPU @ 2.20GHz × 12
图形 GeForce GTX 1060/PCIe/SSE2
操作系统类型 64 位
磁盘 237.3 GB

2、编写 c 应用程序，通过本地 gcc 编译应用程序

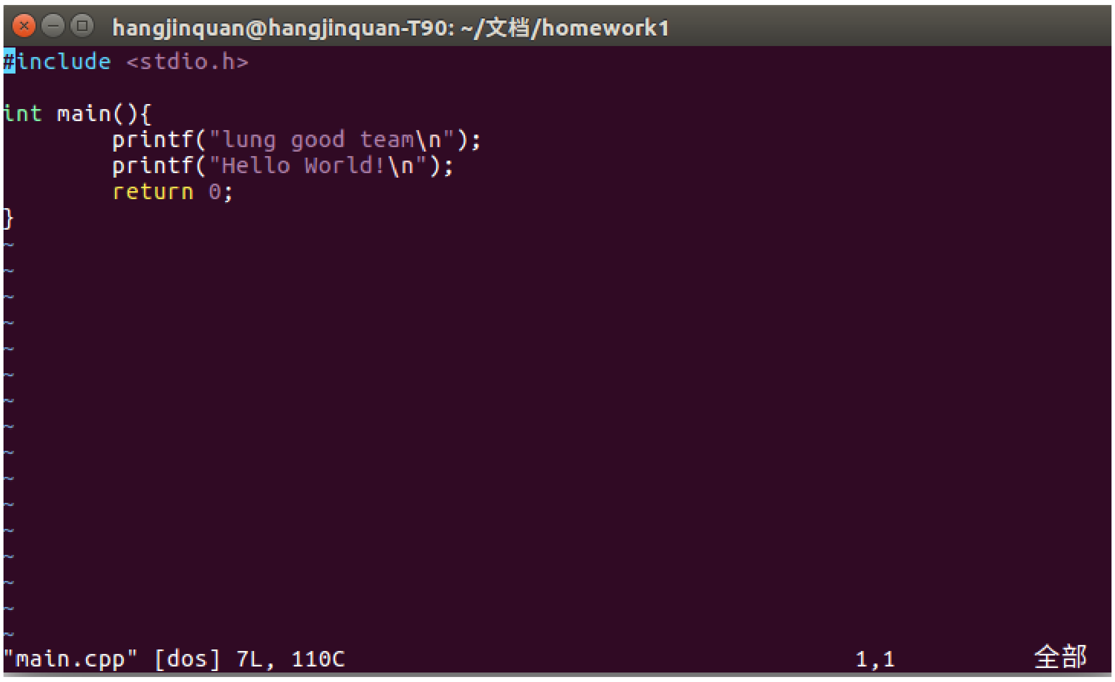
（1）查看编译器的版本号和版权信息

在终端中输入 cc –version 查看编译器信息

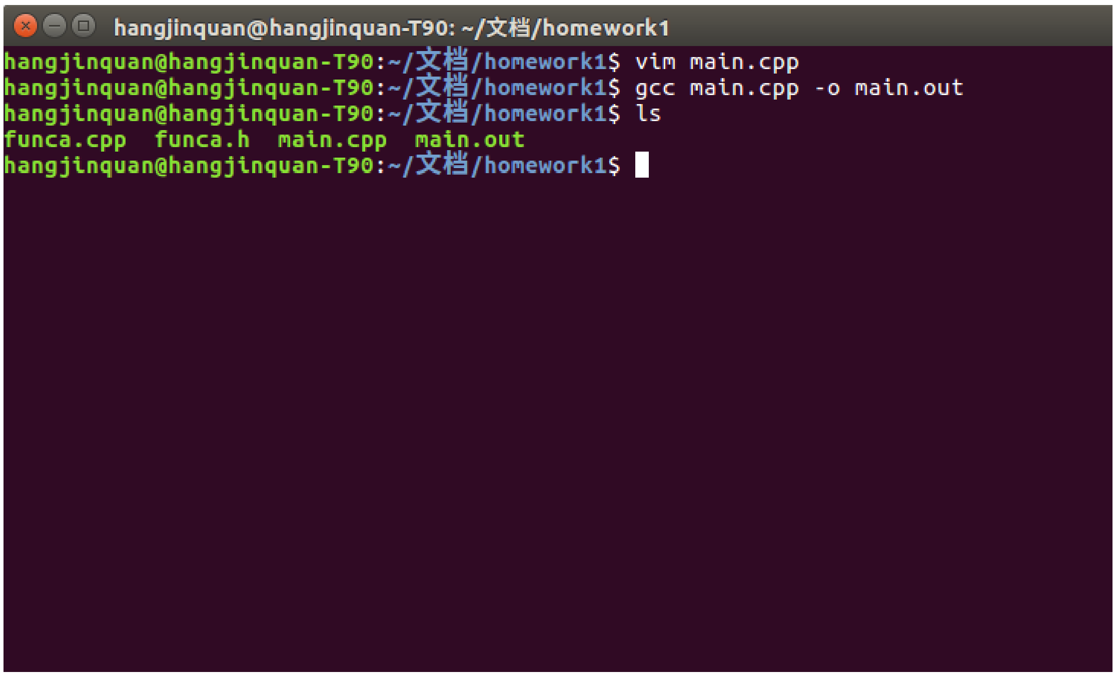

```
sk@ubuntu:~$ cc --version
cc (Ubuntu 5.4.0-6ubuntu1~16.04.12) 5.4.0 20160609
Copyright (C) 2015 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
```

发现已安装版本为 5.4.0 的 gcc 编译器

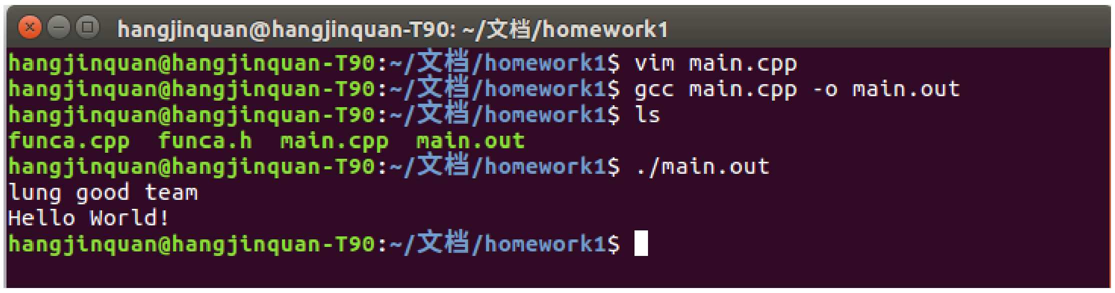
（2）编写 C 应用程序


```
hangjinquan@hangjinquan-T90: ~/文档/homework1
#include <stdio.h>

int main(){
        printf("lung good team\n");
        printf("Hello World!\n");
        return 0;
}
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
"main.cpp" [dos] 7L, 110C                           1,1           全部
```

（3）通过 gcc 编译程序


```
hangjinquan@hangjinquan-T90: ~/文档/homework1
hangjinquan@hangjinquan-T90:~/文档/homework1$ vim main.cpp
hangjinquan@hangjinquan-T90:~/文档/homework1$ gcc main.cpp -o main.out
hangjinquan@hangjinquan-T90:~/文档/homework1$ ls
funca.cpp  funca.h  main.cpp  main.out
hangjinquan@hangjinquan-T90:~/文档/homework1$ 
```

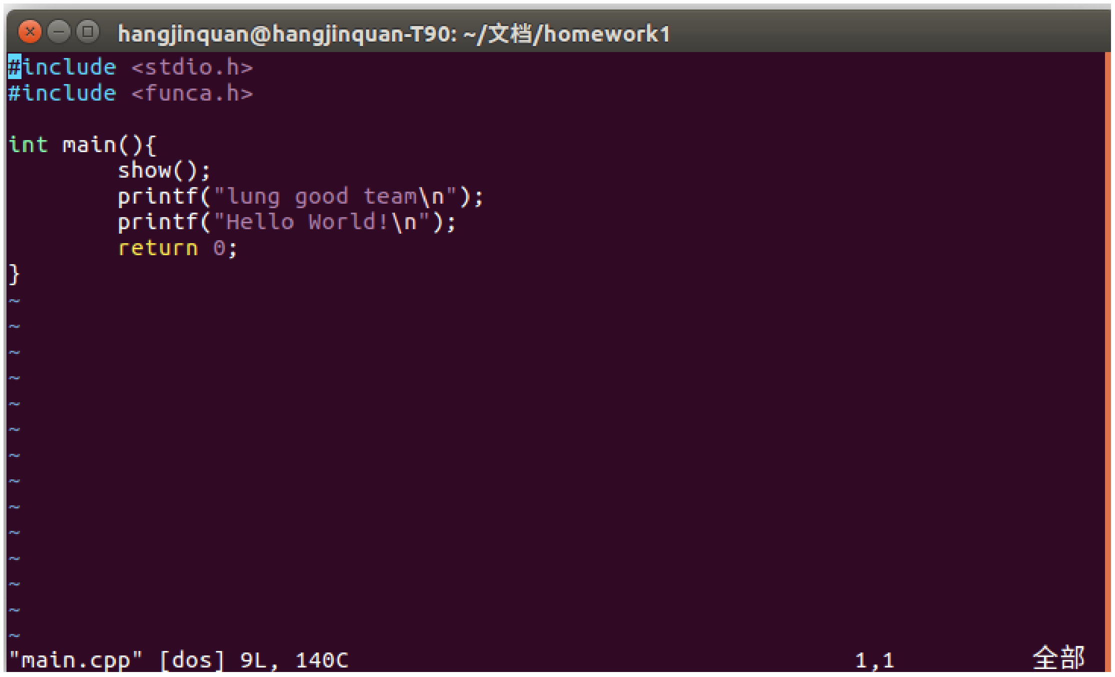
（4）执行程序，查看效果



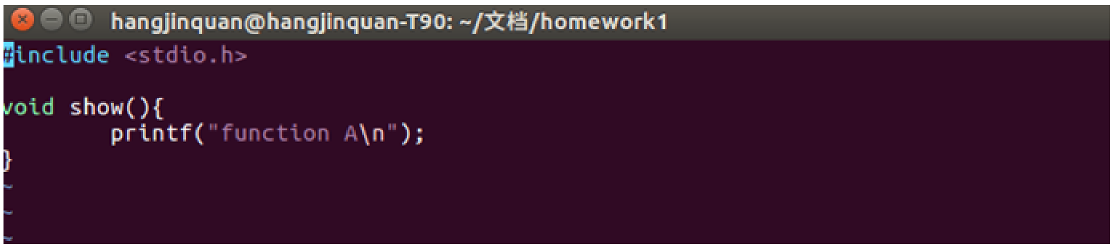
```
hangjinquan@hangjinquan-T90: ~/文档/homework1
hangjinquan@hangjinquan-T90:~/文档/homework1$ vim main.cpp
hangjinquan@hangjinquan-T90:~/文档/homework1$ gcc main.cpp -o main.out
hangjinquan@hangjinquan-T90:~/文档/homework1$ ls
funca.cpp  funca.h  main.cpp  main.out
hangjinquan@hangjinquan-T90:~/文档/homework1$ ./main.out
lung good team
Hello World!
hangjinquan@hangjinquan-T90:~/文档/homework1$ 
```

3、编写 Makefile 管理应用程序，为更好体现 Makefile 的作用，需编写多个 c 程序，给出所创建的 Makefile 的内容；

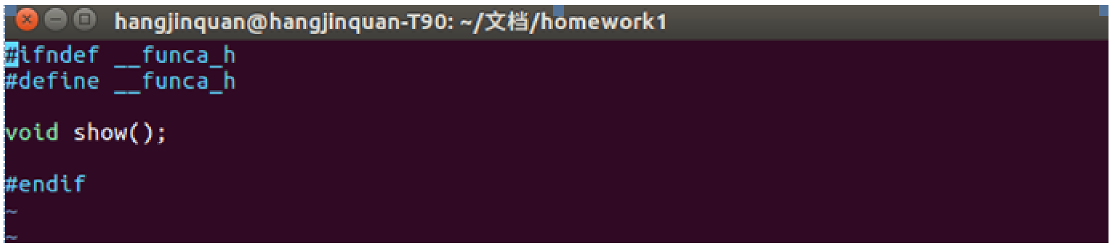
（1）编写 main.c、funca.c 和 funca.h 三个文件，并在 main.c 中调用 funca.c 中的函数



```
hangjinquan@hangjinquan-T90: ~/文档/homework1
#include <stdio.h>
#include <funca.h>

int main(){
        show();
        printf("lung good team\n");
        printf("Hello World!\n");
        return 0;
}
~
~
~
~
~
~
~
~
~
~
~
~
~
~
"main.cpp" [dos] 9L, 140C                                1,1           全部
```

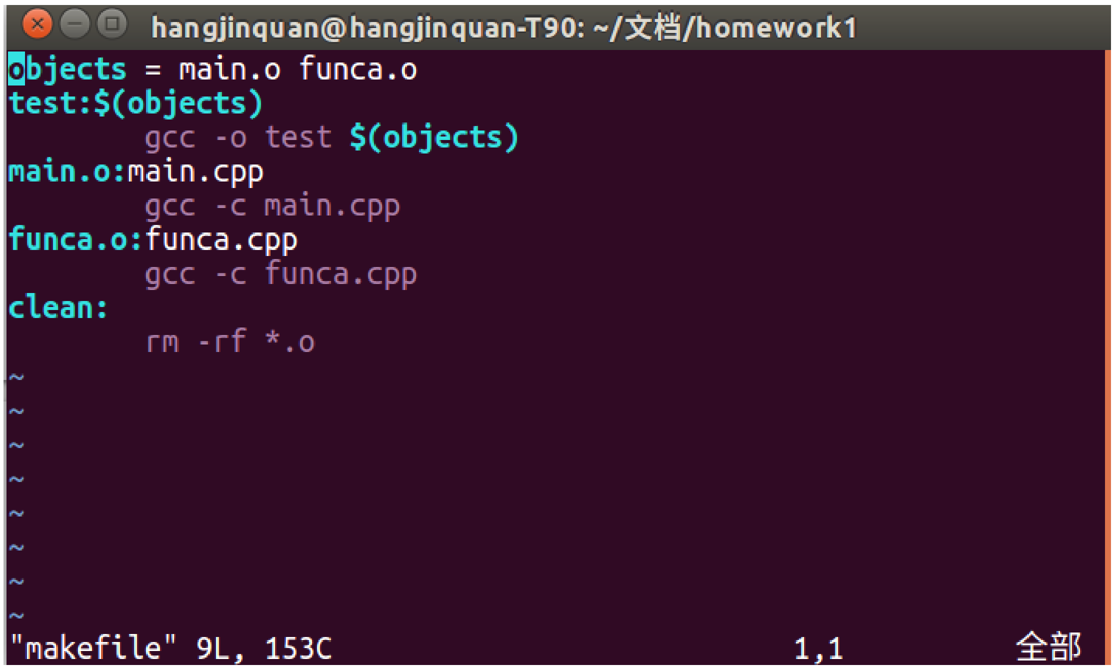


```
hangjinquan@hangjinquan-T90: ~/文档/homework1
#include <stdio.h>

void show(){
        printf("function A\n");
}
~
~
~
```



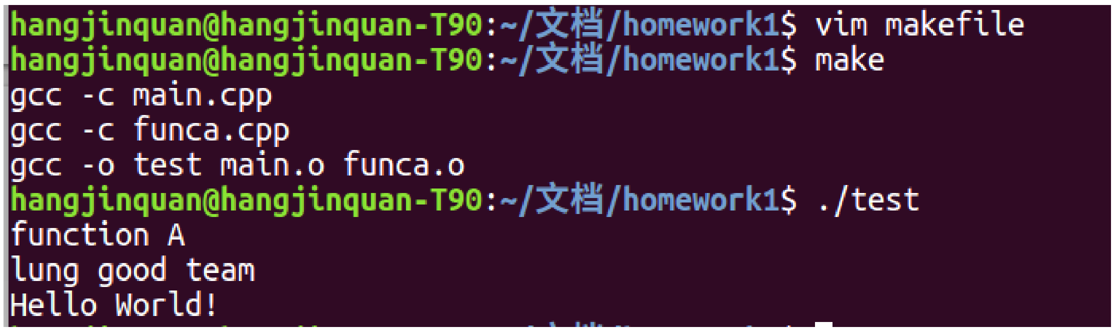
```
hangjinquan@hangjinquan-T90: ~/文档/homework1
#ifndef __funca_h
#define __funca_h

void show();

#endif
~
~
```

（2）编写 makefile 文件



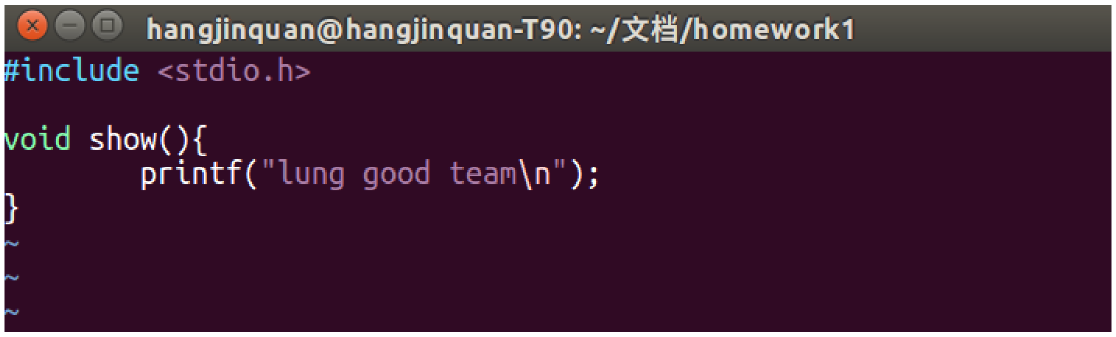
```
hangjinquan@hangjinquan-T90: ~/文档/homework1
objects = main.o funca.o
test:$(objects)
	gcc -o test $(objects)
main.o:main.cpp
	gcc -c main.cpp
funca.o:funca.cpp
	gcc -c funca.cpp
clean:
	rm -rf *.o
~
~
~
~
~
~
~
~
"makefile" 9L, 153C                                   1,1           全部
```

（3）执行 make 操作，进行编译和链接，并执行链接后的程序查看效果



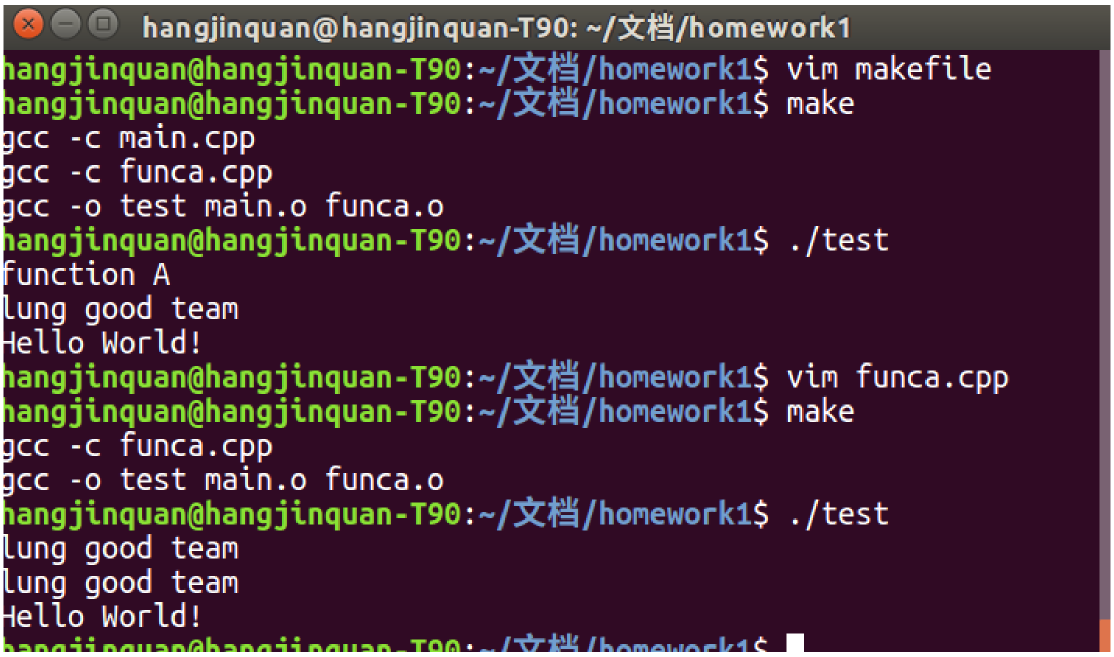
```
hangjinquan@hangjinquan-T90:~/文档/homework1$ vim makefile
hangjinquan@hangjinquan-T90:~/文档/homework1$ make
gcc -c main.cpp
gcc -c funca.cpp
gcc -o test main.o funca.o
hangjinquan@hangjinquan-T90:~/文档/homework1$ ./test
function A
lung good team
Hello World!
```

（5）改变 funca.c 程序中的内容



```
hangjinquan@hangjinquan-T90: ~/文档/homework1
#include <stdio.h>

void show(){
	printf("lung good team\n");
}
~
~
~
```

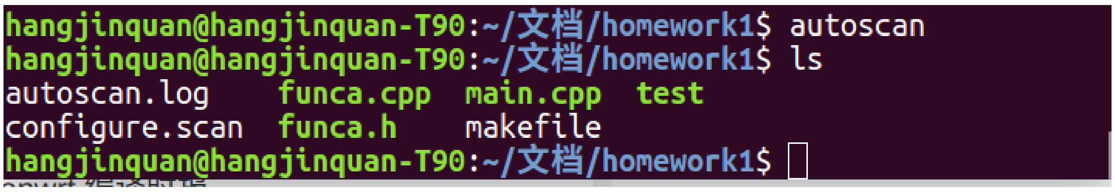
（6）再次用 make 编译文件，然后执行程序查看效果



```
hangjinquan@hangjinquan-T90: ~/文档/homework1
hangjinquan@hangjinquan-T90:~/文档/homework1$ vim makefile
hangjinquan@hangjinquan-T90:~/文档/homework1$ make
gcc -c main.cpp
gcc -c funca.cpp
gcc -o test main.o funca.o
hangjinquan@hangjinquan-T90:~/文档/homework1$ ./test
function A
lung good team
Hello World!
hangjinquan@hangjinquan-T90:~/文档/homework1$ vim funca.cpp
hangjinquan@hangjinquan-T90:~/文档/homework1$ make
gcc -c funca.cpp
gcc -o test main.o funca.o
hangjinquan@hangjinquan-T90:~/文档/homework1$ ./test
lung good team
lung good team
Hello World!
hangjinquan@hangjinquan-T90:~/文档/homework1$
```

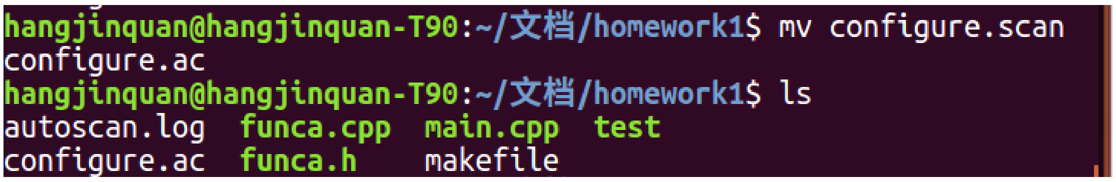
4、通过 autotools 生成 Makefile，完成常用的 make 操作(make, make install, make uninstall, make dist)；

（1）执行 autoscan 命令，生成 configure.sccan 文件

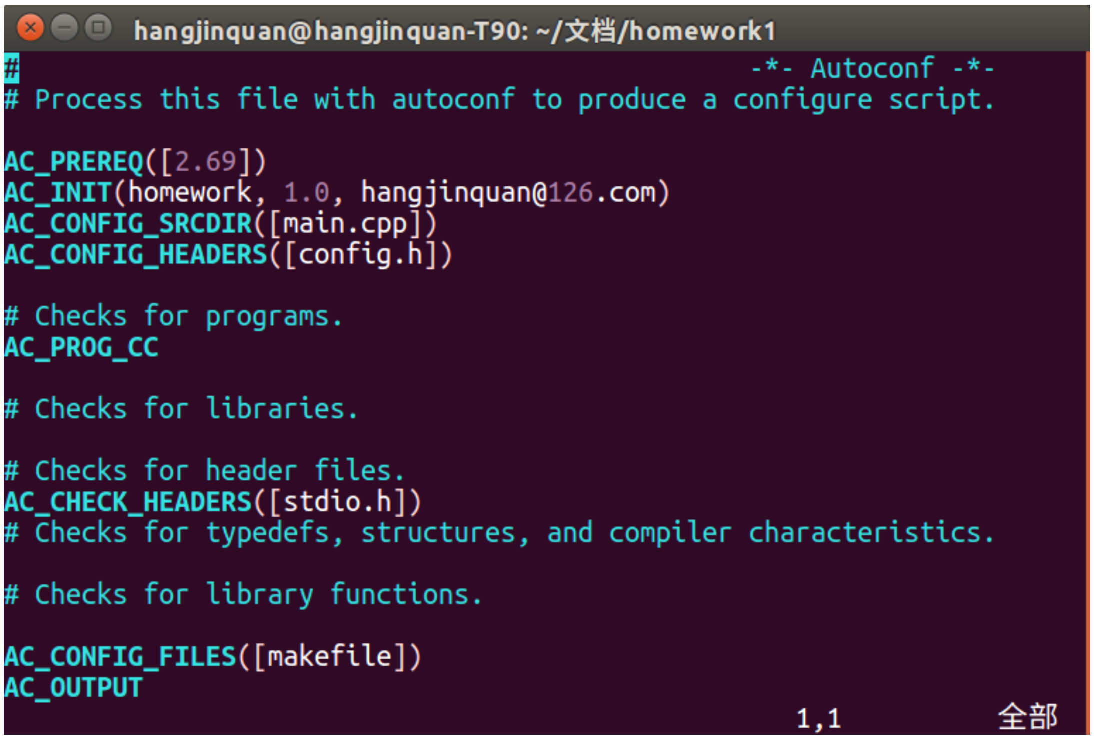


```
hangjinquan@hangjinquan-T90:~/文档/homework1$ autoscan
hangjinquan@hangjinquan-T90:~/文档/homework1$ ls
autoscan.log    funca.cpp  main.cpp  test
configure.scan  funca.h    makefile
hangjinquan@hangjinquan-T90:~/文档/homework1$
```

（2）将 configure.scan 重命名为 configure.ac，并修改其中配置

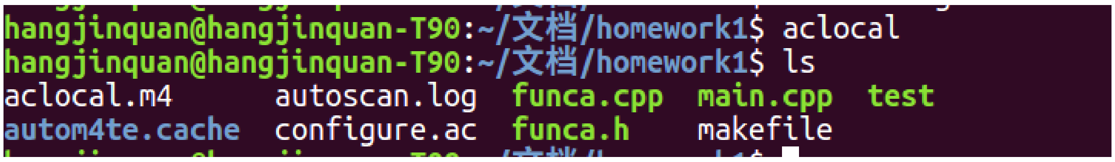


```
hangjinquan@hangjinquan-T90:~/文档/homework1$ mv configure.scan
configure.ac
hangjinquan@hangjinquan-T90:~/文档/homework1$ ls
autoscan.log  funca.cpp  main.cpp  test
configure.ac  funca.h    makefile
```



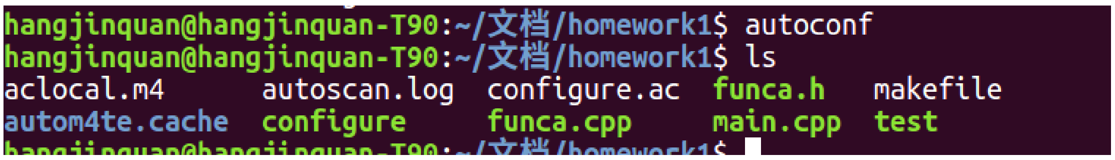
```
hangjinquan@hangjinquan-T90: ~/文档/homework1
#                                               -*- Autoconf -*-
# Process this file with autoconf to produce a configure script.

AC_PREREQ([2.69])
AC_INIT(homework, 1.0, hangjinquan@126.com)
AC_CONFIG_SRCDIR([main.cpp])
AC_CONFIG_HEADERS([config.h])

# Checks for programs.
AC_PROG_CC

# Checks for libraries.

# Checks for header files.
AC_CHECK_HEADERS([stdio.h])
# Checks for typedefs, structures, and compiler characteristics.

# Checks for library functions.

AC_CONFIG_FILES([makefile])
AC_OUTPUT
                                                   1,1          全部
```

（3）执行 aclocal 命令：会扫描 configure.ac 生成 aclocal.mp4 文件



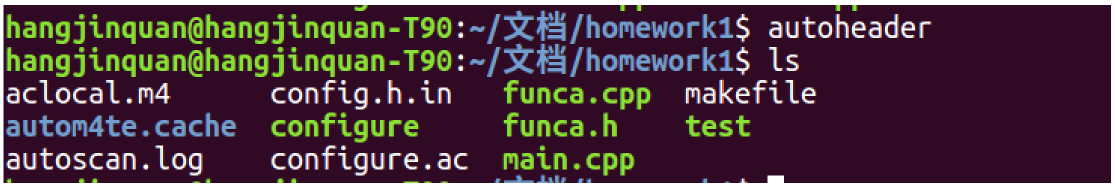
```
hangjinquan@hangjinquan-T90:~/文档/homework1$ aclocal
hangjinquan@hangjinquan-T90:~/文档/homework1$ ls
aclocal.m4       autoscan.log  funca.cpp  main.cpp  test
autom4te.cache   configure.ac  funca.h    makefile
```

（4）执行 autoconf 命令，生成 configure 脚本



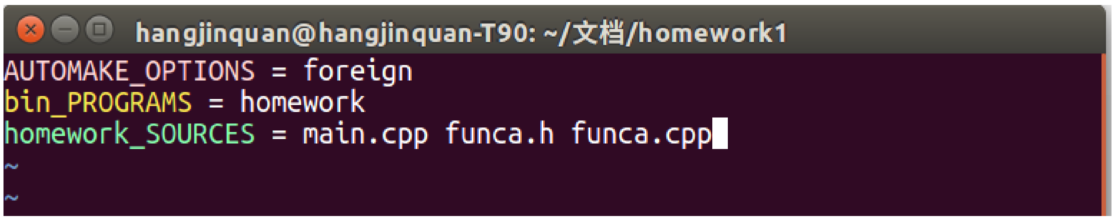
```
hangjinquan@hangjinquan-T90:~/文档/homework1$ autoconf
hangjinquan@hangjinquan-T90:~/文档/homework1$ ls
aclocal.m4      autoscan.log  configure.ac  funca.h   makefile
autom4te.cache  configure     funca.cpp     main.cpp  test
```

（5）执行 autoheader 命令，生成 config.h.in 文件



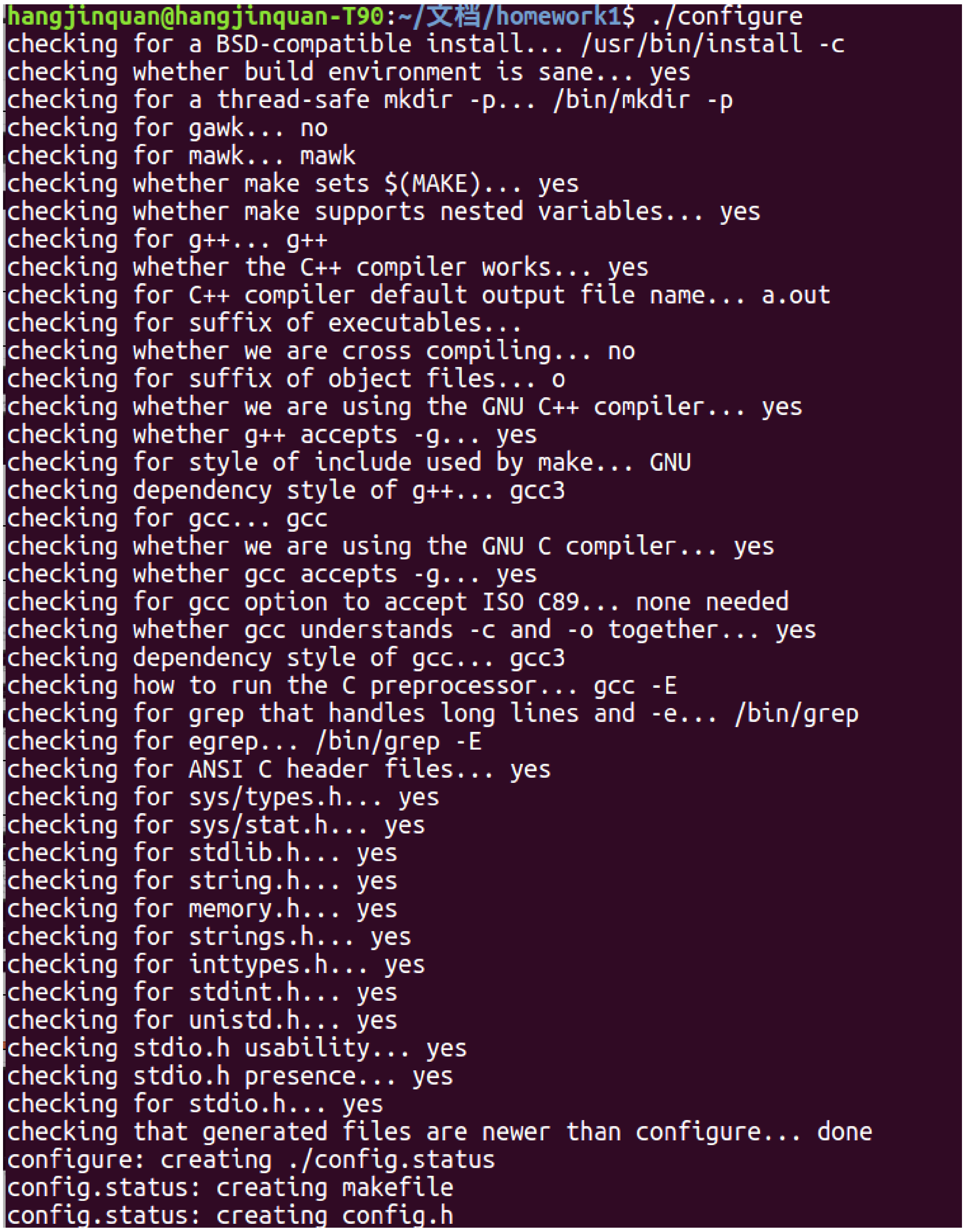
```
hangjinquan@hangjinquan-T90:~/文档/homework1$ autoheader
hangjinquan@hangjinquan-T90:~/文档/homework1$ ls
aclocal.m4      config.h.in   funca.cpp  makefile
autom4te.cache  configure     funca.h    test
autoscan.log    configure.ac  main.cpp
```

(7)创建 makefile.am 文件，定义一些生成 Makefile 的规则



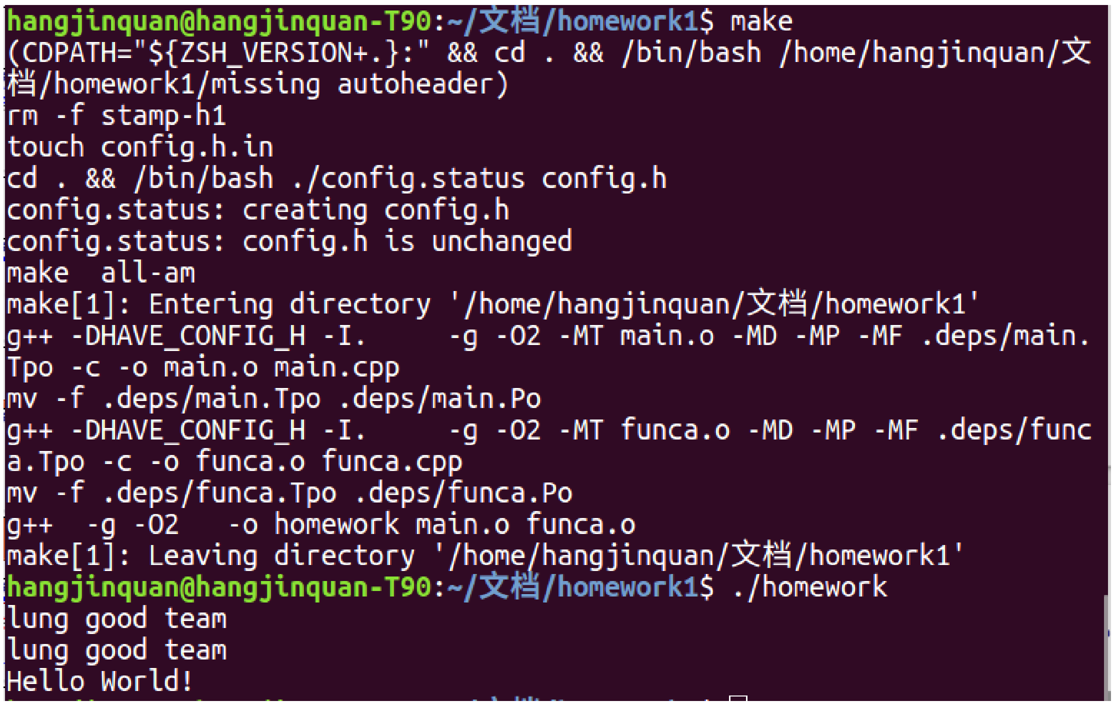
```
hangjinquan@hangjinquan-T90: ~/文档/homework1
AUTOMAKE_OPTIONS = foreign
bin_PROGRAMS = homework
homework_SOURCES = main.cpp funca.h funca.cpp
~
~
```

（8）执行 automake --add-missing 命令，生成 Makefile.in 文件。使用选项"--add-missing" 可以让 Automake 自动添加一些必需的脚本文件。并利用./congigure 把 Makefile.in 变成最终的 Makefile 文件。configure 会把一些配置参数配置到 Makefile 文件里面。



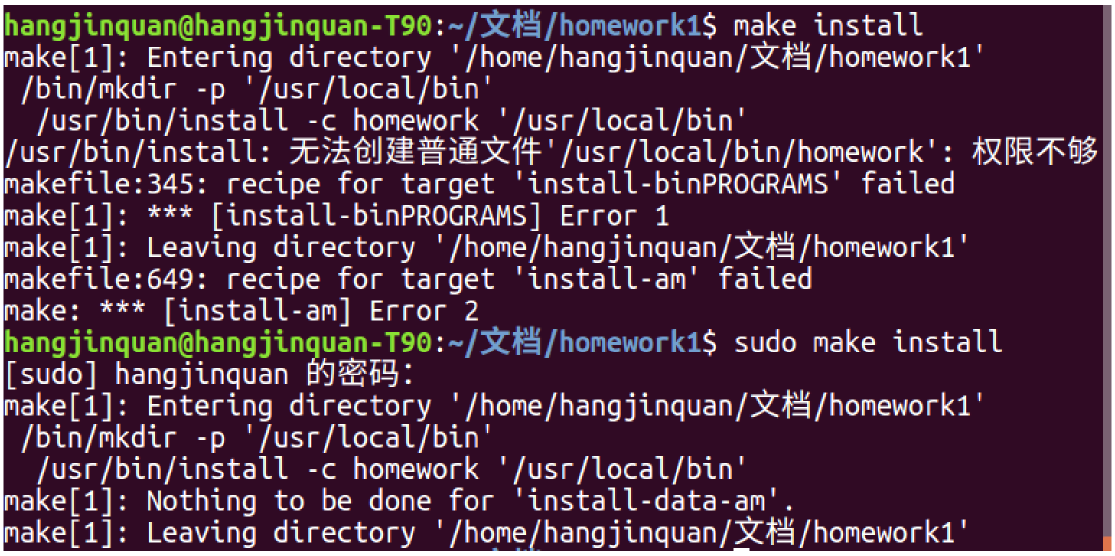
```
hangjinquan@hangjinquan-T90:~/文档/homework1$ ./configure
checking for a BSD-compatible install... /usr/bin/install -c
checking whether build environment is sane... yes
checking for a thread-safe mkdir -p... /bin/mkdir -p
checking for gawk... no
checking for mawk... mawk
checking whether make sets $(MAKE)... yes
checking whether make supports nested variables... yes
checking for g++... g++
checking whether the C++ compiler works... yes
checking for C++ compiler default output file name... a.out
checking for suffix of executables...
checking whether we are cross compiling... no
checking for suffix of object files... o
checking whether we are using the GNU C++ compiler... yes
checking whether g++ accepts -g... yes
checking for style of include used by make... GNU
checking dependency style of g++... gcc3
checking for gcc... gcc
checking whether we are using the GNU C compiler... yes
checking whether gcc accepts -g... yes
checking for gcc option to accept ISO C89... none needed
checking whether gcc understands -c and -o together... yes
checking dependency style of gcc... gcc3
checking how to run the C preprocessor... gcc -E
checking for grep that handles long lines and -e... /bin/grep
checking for egrep... /bin/grep -E
checking for ANSI C header files... yes
checking for sys/types.h... yes
checking for sys/stat.h... yes
checking for stdlib.h... yes
checking for string.h... yes
checking for memory.h... yes
checking for strings.h... yes
checking for inttypes.h... yes
checking for stdint.h... yes
checking for unistd.h... yes
checking stdio.h usability... yes
checking stdio.h presence... yes
checking for stdio.h... yes
checking that generated files are newer than configure... done
configure: creating ./config.status
config.status: creating makefile
config.status: creating config.h
```

（9）执行 make 命令，生成可执行文件 homework，并执行 homework 文件，查看效果

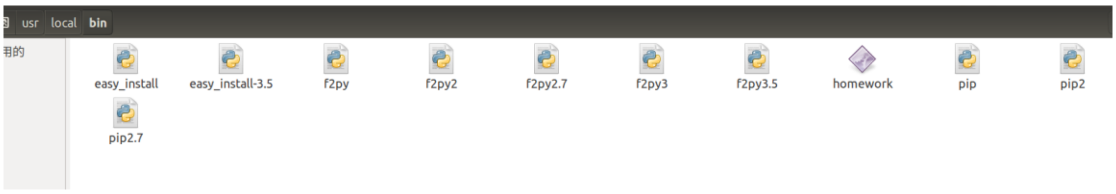


```
hangjinquan@hangjinquan-T90:~/文档/homework1$ make
(CDPATH="${ZSH_VERSION+.}:" && cd . && /bin/bash /home/hangjinquan/文
档/homework1/missing autoheader)
rm -f stamp-h1
touch config.h.in
cd . && /bin/bash ./config.status config.h
config.status: creating config.h
config.status: config.h is unchanged
make  all-am
make[1]: Entering directory '/home/hangjinquan/文档/homework1'
g++ -DHAVE_CONFIG_H -I.     -g -O2 -MT main.o -MD -MP -MF .deps/main.
Tpo -c -o main.o main.cpp
mv -f .deps/main.Tpo .deps/main.Po
g++ -DHAVE_CONFIG_H -I.     -g -O2 -MT funca.o -MD -MP -MF .deps/func
a.Tpo -c -o funca.o funca.cpp
mv -f .deps/funca.Tpo .deps/funca.Po
g++  -g -O2   -o homework main.o funca.o
make[1]: Leaving directory '/home/hangjinquan/文档/homework1'
hangjinquan@hangjinquan-T90:~/文档/homework1$ ./homework
lung good team
lung good team
Hello World!
```

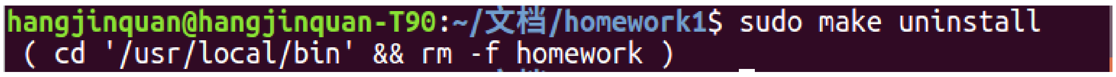
（10）执行 make install 操作，可以看到可执行文件被拷贝到了/usr/local/bin 中



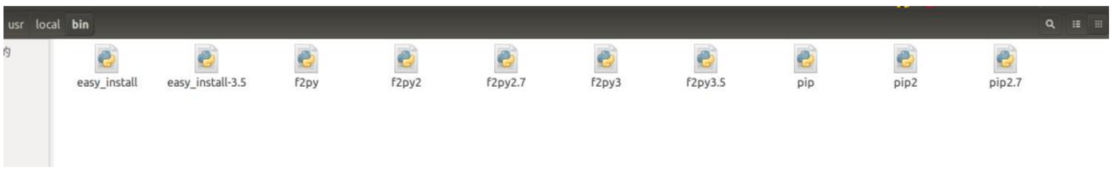
```
hangjinquan@hangjinquan-T90:~/文档/homework1$ make install
make[1]: Entering directory '/home/hangjinquan/文档/homework1'
 /bin/mkdir -p '/usr/local/bin'
  /usr/bin/install -c homework '/usr/local/bin'
/usr/bin/install: 无法创建普通文件'/usr/local/bin/homework': 权限不够
makefile:345: recipe for target 'install-binPROGRAMS' failed
make[1]: *** [install-binPROGRAMS] Error 1
make[1]: Leaving directory '/home/hangjinquan/文档/homework1'
makefile:649: recipe for target 'install-am' failed
make: *** [install-am] Error 2
hangjinquan@hangjinquan-T90:~/文档/homework1$ sudo make install
[sudo] hangjinquan 的密码:
make[1]: Entering directory '/home/hangjinquan/文档/homework1'
 /bin/mkdir -p '/usr/local/bin'
  /usr/bin/install -c homework '/usr/local/bin'
make[1]: Nothing to be done for 'install-data-am'.
make[1]: Leaving directory '/home/hangjinquan/文档/homework1'
```



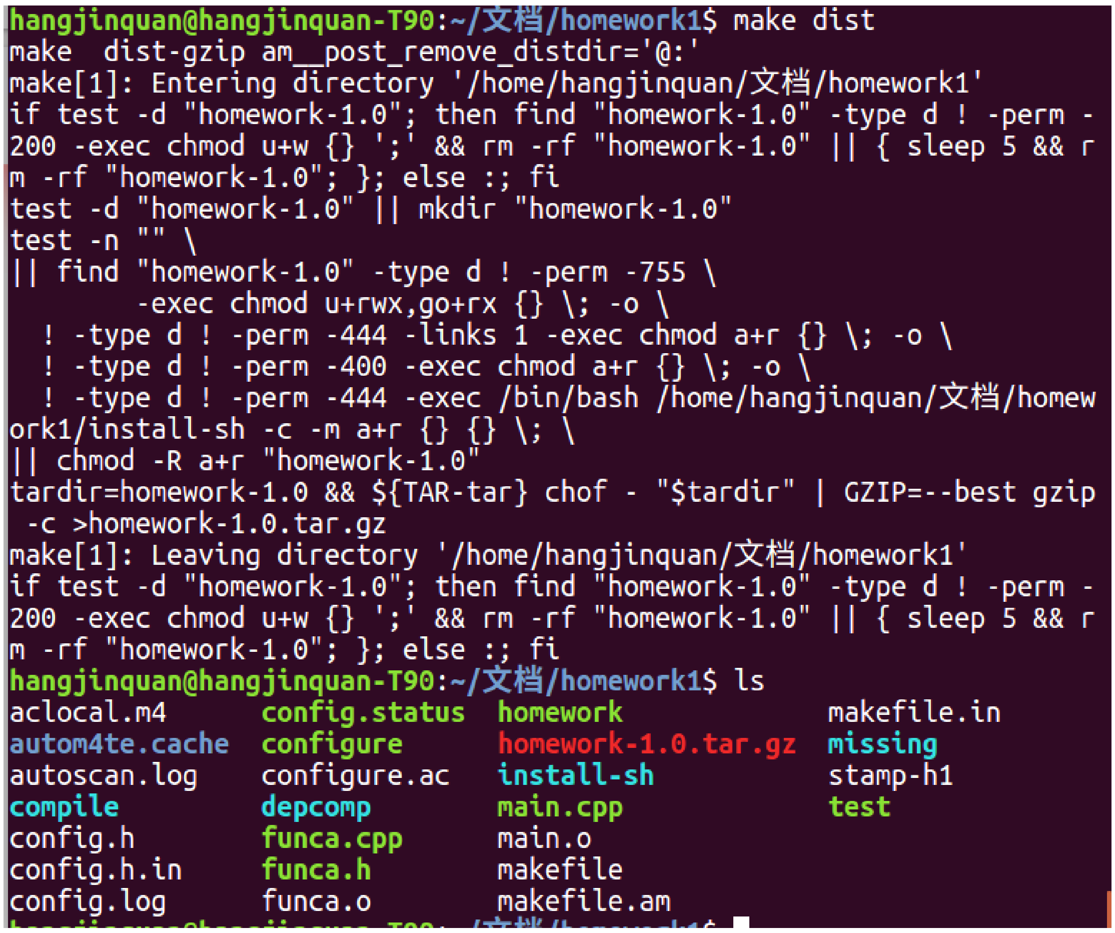
（10）执行 make uninstall 操作，可以看到可执行文件从系统目录中被卸载了



```
hangjinquan@hangjinquan-T90:~/文档/homework1$ sudo make uninstall
 ( cd '/usr/local/bin' && rm -f homework )
```



（11）执行 make dist 操作，这个命令将会将可执行文件及相关文件打包成一个 tar.gz 压缩的文件用来作为发布软件的软件包。可以看到文件夹中生成了 homework-1.0.tar.gz 文件

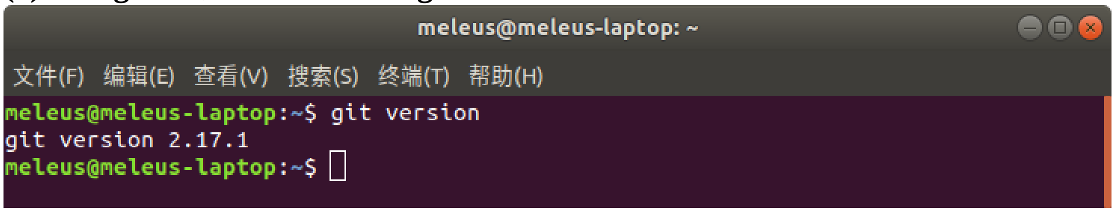


```
hangjinquan@hangjinquan-T90:~/文档/homework1$ make dist
make  dist-gzip am__post_remove_distdir='@:'
make[1]: Entering directory '/home/hangjinquan/文档/homework1'
if test -d "homework-1.0"; then find "homework-1.0" -type d ! -perm -
200 -exec chmod u+w {} ';' && rm -rf "homework-1.0" || { sleep 5 && r
m -rf "homework-1.0"; }; else :; fi
test -d "homework-1.0" || mkdir "homework-1.0"
test -n "" \
|| find "homework-1.0" -type d ! -perm -755 \
        -exec chmod u+rwx,go+rx {} \; -o \
  ! -type d ! -perm -444 -links 1 -exec chmod a+r {} \; -o \
  ! -type d ! -perm -400 -exec chmod a+r {} \; -o \
  ! -type d ! -perm -444 -exec /bin/bash /home/hangjinquan/文档/homew
ork1/install-sh -c -m a+r {} {} \; \
|| chmod -R a+r "homework-1.0"
tardir=homework-1.0 && ${TAR-tar} chof - "$tardir" | GZIP=--best gzip
 -c >homework-1.0.tar.gz
make[1]: Leaving directory '/home/hangjinquan/文档/homework1'
if test -d "homework-1.0"; then find "homework-1.0" -type d ! -perm -
200 -exec chmod u+w {} ';' && rm -rf "homework-1.0" || { sleep 5 && r
m -rf "homework-1.0"; }; else :; fi
hangjinquan@hangjinquan-T90:~/文档/homework1$ ls
aclocal.m4      config.status  homework             makefile.in
autom4te.cache  configure      homework-1.0.tar.gz  missing
autoscan.log    configure.ac   install-sh           stamp-h1
compile         depcomp        main.cpp             test
config.h        funca.cpp      main.o
config.h.in     funca.h        makefile
config.log      funca.o        makefile.am
```

5、创建小组 git 仓库，github 账号，用来存储小组工作文件以及小组报告；学习如何构建 github 文件，如何上传和下载 github 文件等。

A、与 github 连接

(1)显示 git 版本（sudo install git）



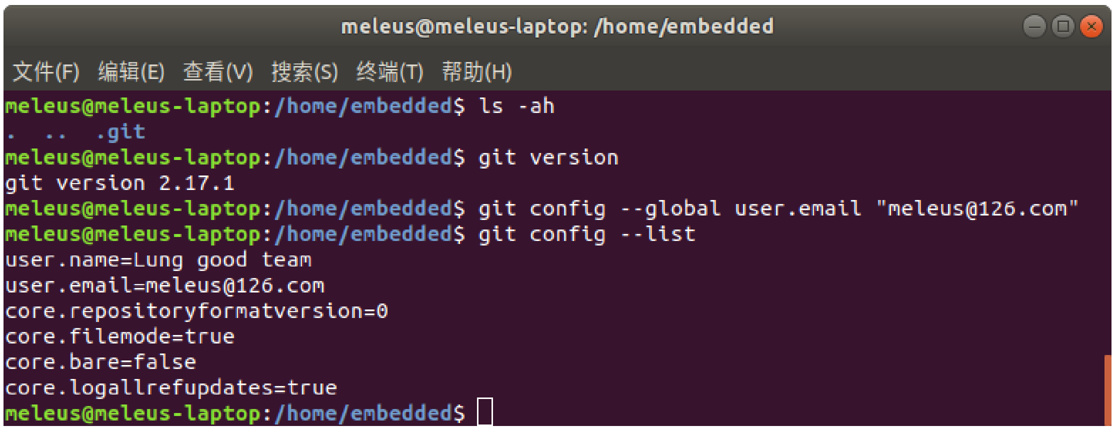
```
meleus@meleus-laptop:~$ git version
git version 2.17.1
meleus@meleus-laptop:~$ 
```

(2)配置用户名和用户邮件

git config --global user.name "XXXX"

git config --global user.email "XXXXXXXXXX"

git config --list (查看是否配置成功)

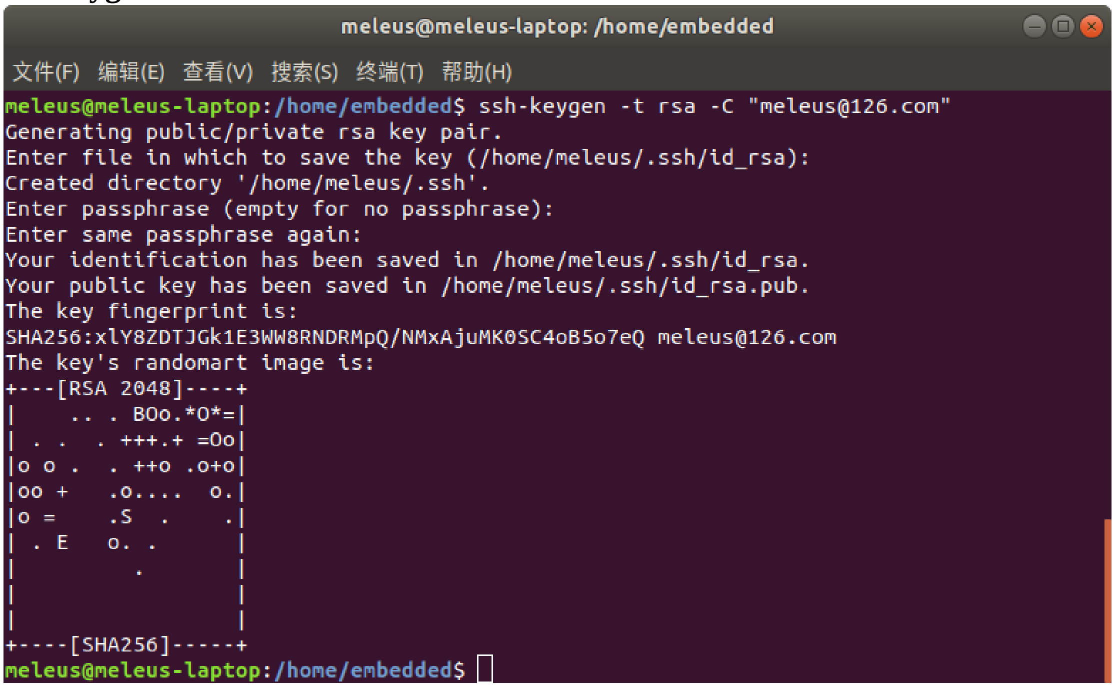


meleus@meleus-laptop: /home/embedded
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)

```
meleus@meleus-laptop:/home/embedded$ ls -ah
.  ..  .git
meleus@meleus-laptop:/home/embedded$ git version
git version 2.17.1
meleus@meleus-laptop:/home/embedded$ git config --global user.email "meleus@126.com"
meleus@meleus-laptop:/home/embedded$ git config --list
user.name=Lung good team
user.email=meleus@126.com
core.repositoryformatversion=0
core.filemode=true
core.bare=false
core.logallrefupdates=true
meleus@meleus-laptop:/home/embedded$ 
```

(3)配置 Git 的私钥和公钥

ssh-keygen -t rsa   -C "用户邮箱"



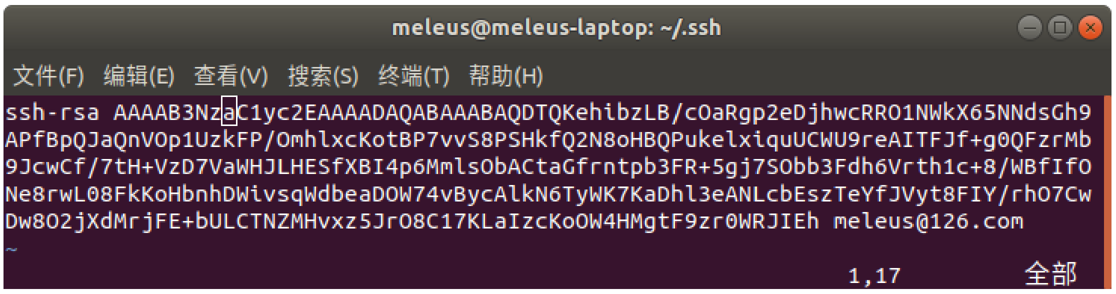
meleus@meleus-laptop: /home/embedded
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)

```
meleus@meleus-laptop:/home/embedded$ ssh-keygen -t rsa -C "meleus@126.com"
Generating public/private rsa key pair.
Enter file in which to save the key (/home/meleus/.ssh/id_rsa):
Created directory '/home/meleus/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/meleus/.ssh/id_rsa.
Your public key has been saved in /home/meleus/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:xlY8ZDTJGk1E3WW8RNDRMpQ/NMxAjuMK0SC4oB5o7eQ meleus@126.com
The key's randomart image is:
+---[RSA 2048]----+
|    .. . BOo.*O*=|
| . .  . +++.+ =Oo|
|o o .  . ++o .o+o|
|oo +   .o....  o.|
|o =    .S  .    .|
| . E   o. .      |
|          .      |
|                 |
|                 |
+----[SHA256]-----+
meleus@meleus-laptop:/home/embedded$ 
```

(4)vim 打开 Git 的公钥证书.

cd ~/.ssh/

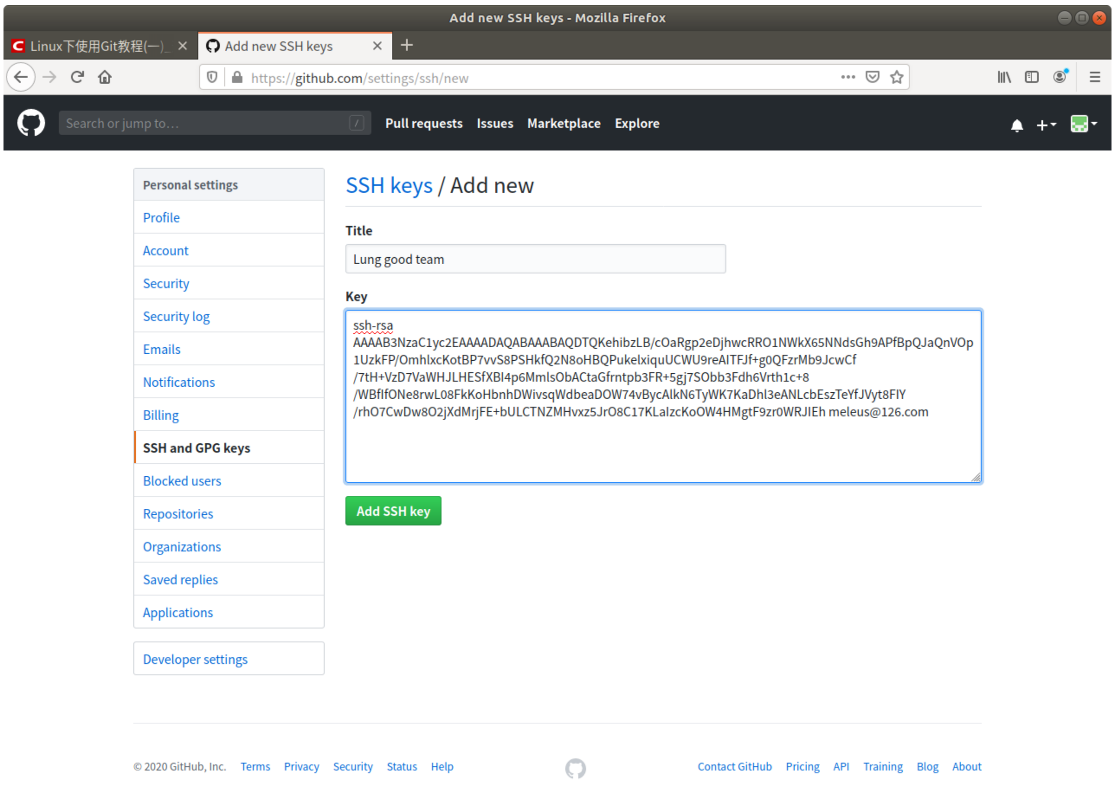
vim id_rsa.pub



meleus@meleus-laptop: ~/.ssh
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)

```
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQDTQKehibzLB/cOaRgp2eDjhwcRRO1NWkX65NNdsGh9
APfBpQJaQnVOp1UzkFP/OmhlxcKotBP7vvS8PSHkfQ2N8oHBQPukelxiquUCWU9reAITFJf+g0QFzrMb
9JcwCf/7tH+VzD7VaWHJLHESfXBI4p6MmlsObACtaGfrntpb3FR+5gj7SObb3Fdh6Vrth1c+8/WBfIfO
Ne8rwL08FkKoHbnhDWivsqWdbeaDOW74vBycAlkN6TyWK7KaDhl3eANLcbEszTeYfJVyt8FIY/rhO7Cw
Dw8O2jXdMrjFE+bULCTNZMHvxz5JrO8C17KLaIzcKoOW4HMgtF9zr0WRJIEh meleus@126.com
~
                                                          1,17          全部
```

(5)粘贴到 github settings 里的 SSH keys



SSH keys / Add new

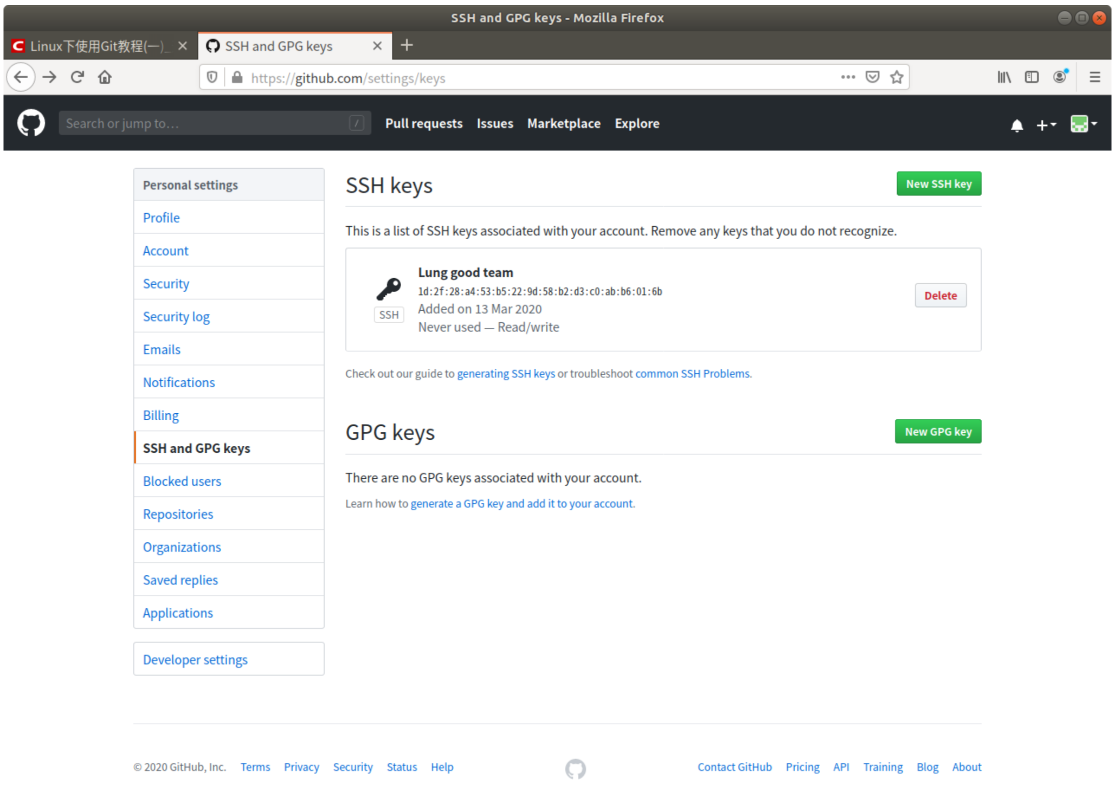
Title

Lung good team

Key

```
ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAABAQDTQKehibzLB/cOaRgp2eDjhwcRRO1NWkX65NNdsGh9APfBpQJaQnVOp
1UzkFP/OmhlxcKotBP7vvS8PSHkfQ2N8oHBQPukelxiquUCWU9reAITFJf+g0QFzrMb9JcwCf
/7tH+VzD7VaWHJLHESfXBI4p6MmlsObACtaGfrntpb3FR+5gj7SObb3Fdh6Vrth1c+8
/WBfIfONe8rwL08FkKoHbnhDWivsqWdbeaDOW74vBycAlkN6TyWK7KaDhl3eANLcbEszTeYfJVyt8FIY
/rhO7CwDw8O2jXdMrjFE+bULCTNZMHvxz5JrO8C17KLalzcKoOW4HMgtF9zr0WRJIEh meleus@126.com
```

完成后是这样



SSH keys

This is a list of SSH keys associated with your account. Remove any keys that you do not recognize.

Lung good team
1d:2f:28:a4:53:b5:22:9d:58:b2:d3:c0:ab:b6:01:6b
Added on 13 Mar 2020
Never used — Read/write

Check out our guide to generating SSH keys or troubleshoot common SSH Problems.

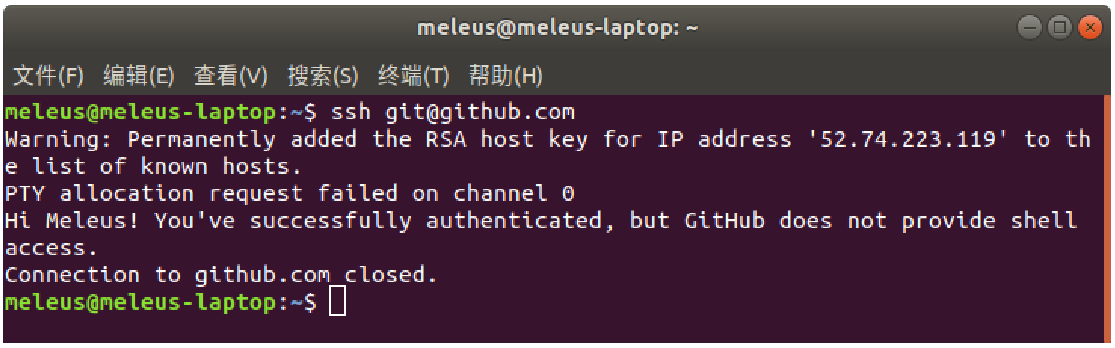
GPG keys

There are no GPG keys associated with your account.

Learn how to generate a GPG key and add it to your account.

(6)验证 Git 是否能够配置.

ssh git@github.com

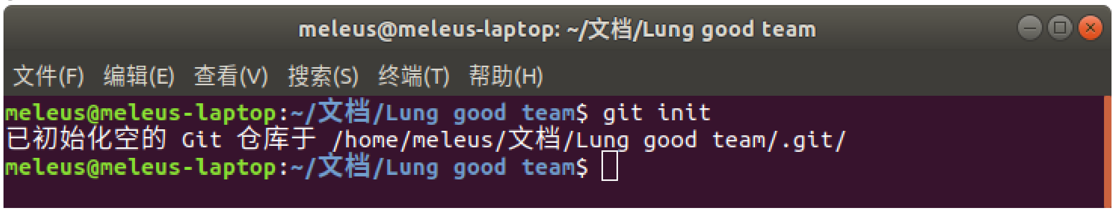


```
meleus@meleus-laptop:~$ ssh git@github.com
Warning: Permanently added the RSA host key for IP address '52.74.223.119' to th
e list of known hosts.
PTY allocation request failed on channel 0
Hi Meleus! You've successfully authenticated, but GitHub does not provide shell
access.
Connection to github.com closed.
meleus@meleus-laptop:~$ 
```

B、本地构建 git 版本库

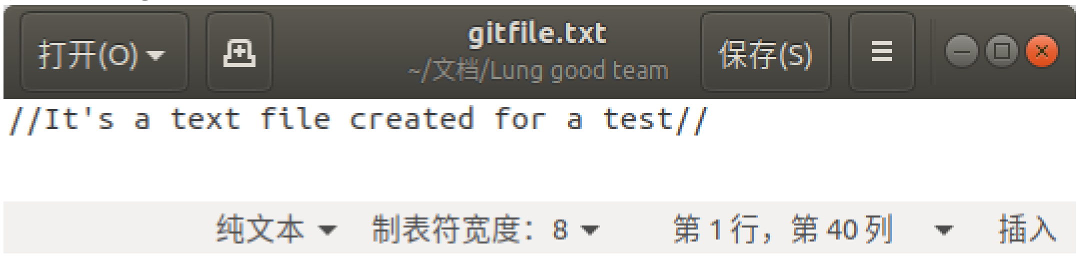
(1)转到你希望的位置初始化版本库

git init


meleus@meleus-laptop: ~/文档/Lung good team
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)

```
meleus@meleus-laptop:~/文档/Lung good team$ git init
已初始化空的 Git 仓库于 /home/meleus/文档/Lung good team/.git/
meleus@meleus-laptop:~/文档/Lung good team$ 
```

(2)创建文件 gitfile.txt



```
//It's a text file created for a test//
```

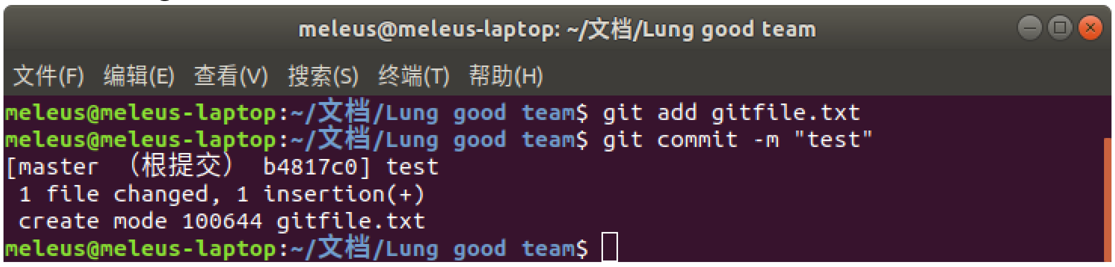
(3)添加进版本库

git add gitfile.txt

git commit -m “comment here”

注意：1、多次 add，而 commit 仅需 1 次（缓冲区&工作区）

2、git commit 后的-m “”为对每个版本的说明


meleus@meleus-laptop: ~/文档/Lung good team
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)

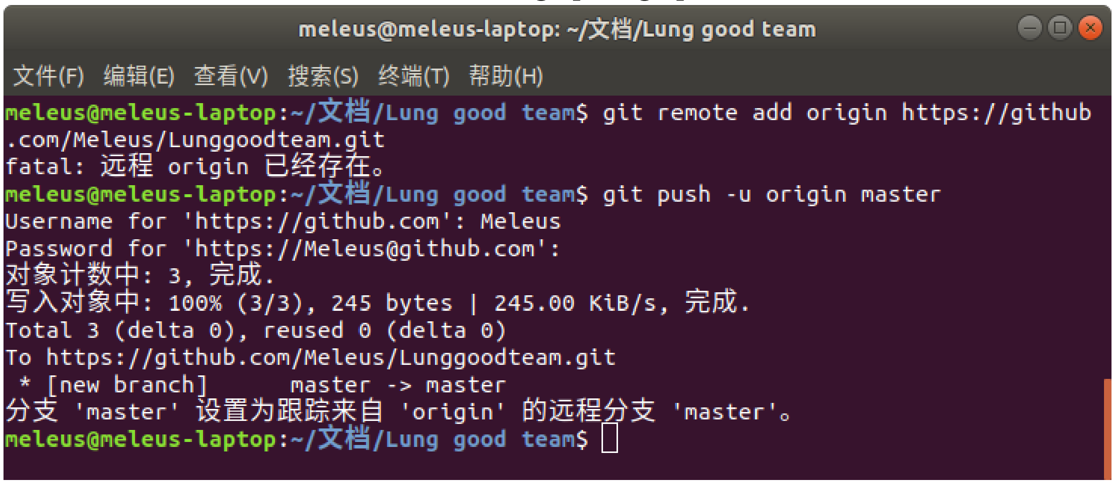
```
meleus@meleus-laptop:~/文档/Lung good team$ git add gitfile.txt
meleus@meleus-laptop:~/文档/Lung good team$ git commit -m "test"
[master （根提交） b4817c0] test
 1 file changed, 1 insertion(+)
 create mode 100644 gitfile.txt
meleus@meleus-laptop:~/文档/Lung good team$ 
```

C、远程仓库管理

(1)GitHub上建立远程仓库后在将本地仓库中的文件push到该远程仓库之前需要关联本地仓库和远程仓库

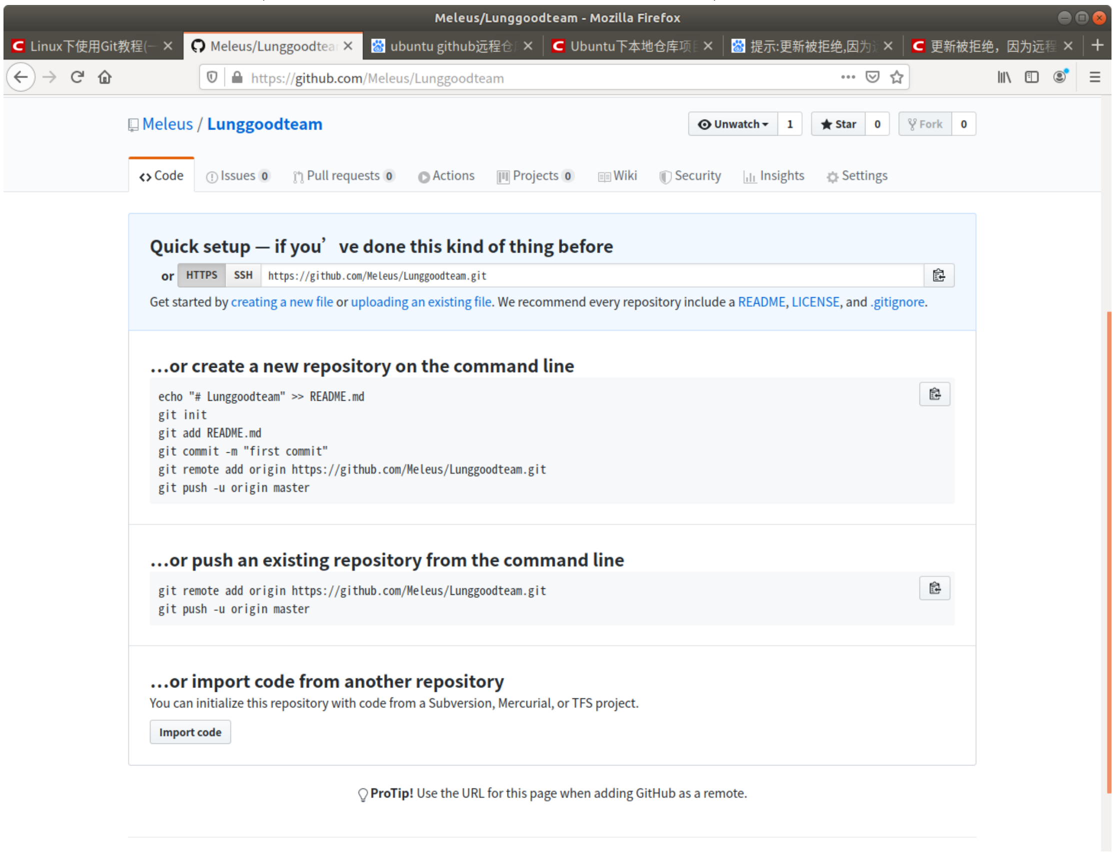
git remote add origin https://github.com/Meleus/Lunggoodteam.git

再执行 git push -u origin master
只有第一次需要加上-u的参数，之后只需要git push/git pull即可


meleus@meleus-laptop: ~/文档/Lung good team
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)

```
meleus@meleus-laptop:~/文档/Lung good team$ git remote add origin https://github
.com/Meleus/Lunggoodteam.git
fatal: 远程 origin 已经存在。
meleus@meleus-laptop:~/文档/Lung good team$ git push -u origin master
Username for 'https://github.com': Meleus
Password for 'https://Meleus@github.com':
对象计数中: 3, 完成.
写入对象中: 100% (3/3), 245 bytes | 245.00 KiB/s, 完成.
Total 3 (delta 0), reused 0 (delta 0)
To https://github.com/Meleus/Lunggoodteam.git
 * [new branch]      master -> master
分支 'master' 设置为跟踪来自 'origin' 的远程分支 'master'。
meleus@meleus-laptop:~/文档/Lung good team$
```

原本新建的远程仓库(不要用网页上传文件，会报引用错误)



Quick setup — if you've done this kind of thing before

or HTTPS SSH https://github.com/Meleus/Lunggoodteam.git

Get started by creating a new file or uploading an existing file. We recommend every repository include a README, LICENSE, and .gitignore.

…or create a new repository on the command line

```
echo "# Lunggoodteam" >> README.md
git init
git add README.md
git commit -m "first commit"
git remote add origin https://github.com/Meleus/Lunggoodteam.git
git push -u origin master
```

…or push an existing repository from the command line

```
git remote add origin https://github.com/Meleus/Lunggoodteam.git
git push -u origin master
```

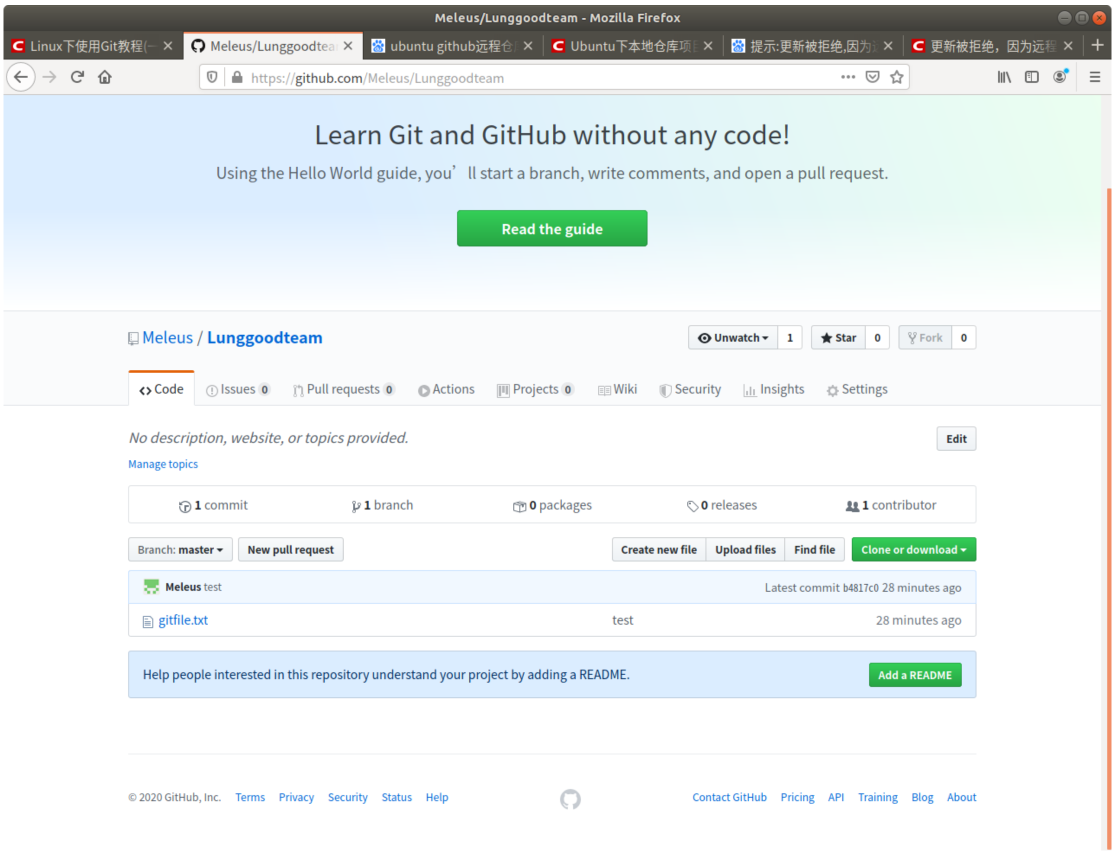
…or import code from another repository

You can initialize this repository with code from a Subversion, Mercurial, or TFS project.

Import code

ProTip! Use the URL for this page when adding GitHub as a remote.

执行完后 gitfile.txt 会被添加进去


Meleus/Lunggoodteam - Mozilla Firefox

https://github.com/Meleus/Lunggoodteam

Learn Git and GitHub without any code!

Using the Hello World guide, you' ll start a branch, write comments, and open a pull request.

Read the guide

Meleus / Lunggoodteam

Unwatch 1 | Star 0 | Fork 0

Code | Issues 0 | Pull requests 0 | Actions | Projects 0 | Wiki | Security | Insights | Settings

No description, website, or topics provided.

Manage topics

1 commit | 1 branch | 0 packages | 0 releases | 1 contributor

Branch: master | New pull request | Create new file | Upload files | Find file | Clone or download

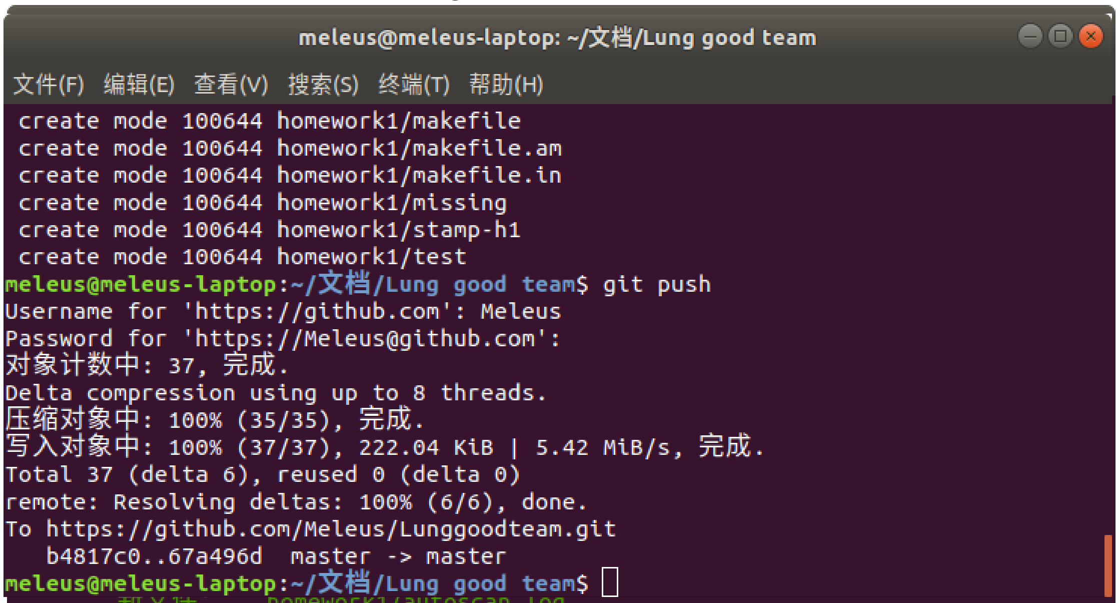
Meleus test — Latest commit b4817c0 28 minutes ago

gitfile.txt	test	28 minutes ago

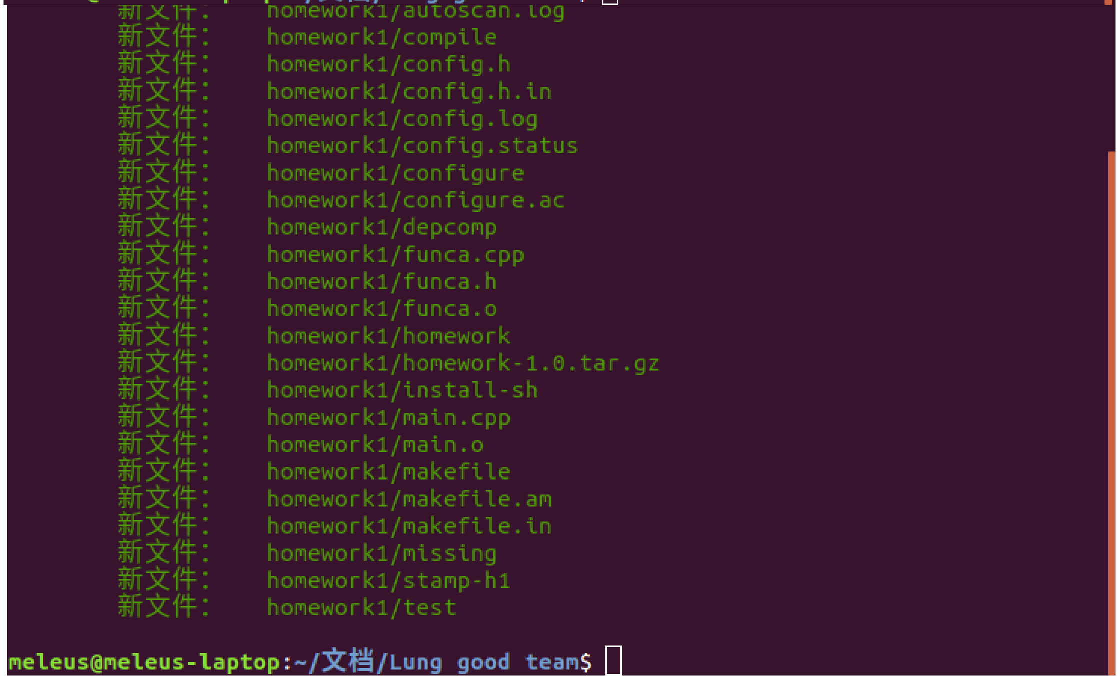
Help people interested in this repository understand your project by adding a README. — Add a README

© 2020 GitHub, Inc. Terms Privacy Security Status Help Contact GitHub Pricing API Training Blog About

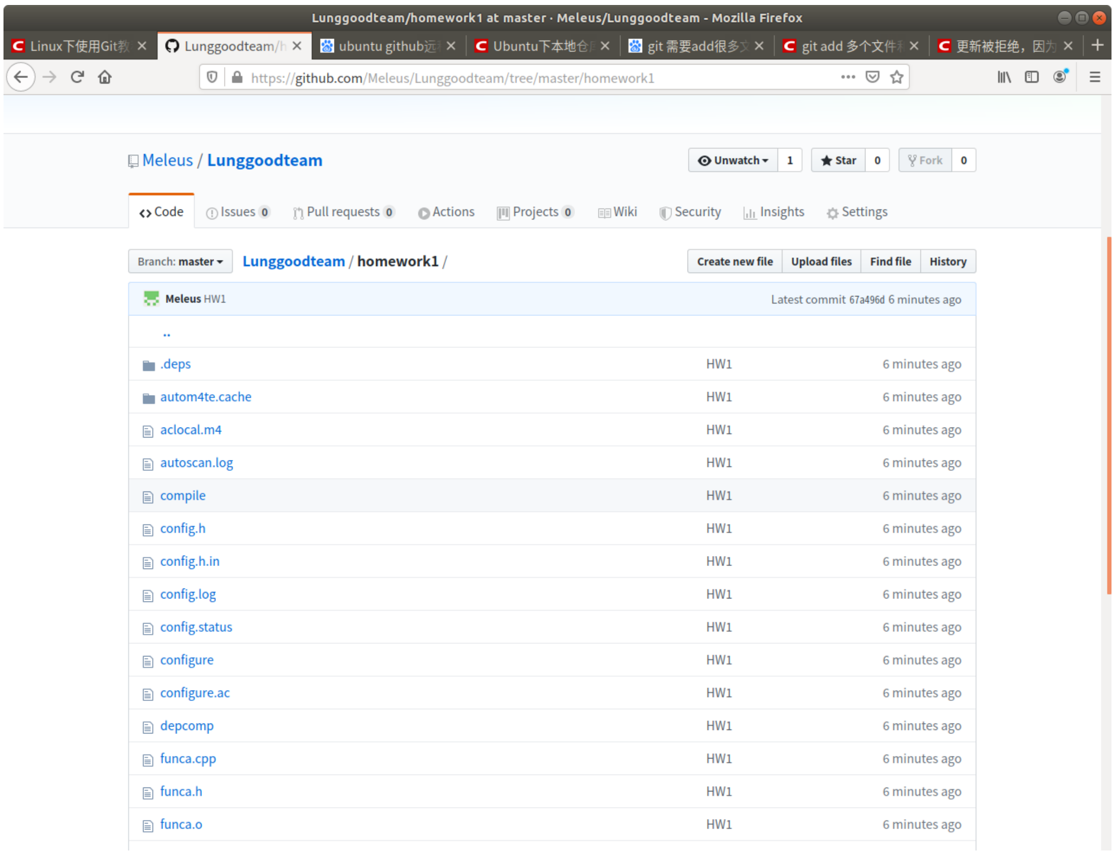
（2）把之前的 makefile 等代码放到 git 里



```
 create mode 100644 homework1/makefile
 create mode 100644 homework1/makefile.am
 create mode 100644 homework1/makefile.in
 create mode 100644 homework1/missing
 create mode 100644 homework1/stamp-h1
 create mode 100644 homework1/test
meleus@meleus-laptop:~/文档/Lung good team$ git push
Username for 'https://github.com': Meleus
Password for 'https://Meleus@github.com':
对象计数中: 37, 完成.
Delta compression using up to 8 threads.
压缩对象中: 100% (35/35), 完成.
写入对象中: 100% (37/37), 222.04 KiB | 5.42 MiB/s, 完成.
Total 37 (delta 6), reused 0 (delta 0)
remote: Resolving deltas: 100% (6/6), done.
To https://github.com/Meleus/Lunggoodteam.git
   b4817c0..67a496d  master -> master
meleus@meleus-laptop:~/文档/Lung good team$ 
```



```
        新文件:    homework1/autoscan.log
        新文件:    homework1/compile
        新文件:    homework1/config.h
        新文件:    homework1/config.h.in
        新文件:    homework1/config.log
        新文件:    homework1/config.status
        新文件:    homework1/configure
        新文件:    homework1/configure.ac
        新文件:    homework1/depcomp
        新文件:    homework1/funca.cpp
        新文件:    homework1/funca.h
        新文件:    homework1/funca.o
        新文件:    homework1/homework
        新文件:    homework1/homework-1.0.tar.gz
        新文件:    homework1/install-sh
        新文件:    homework1/main.cpp
        新文件:    homework1/main.o
        新文件:    homework1/makefile
        新文件:    homework1/makefile.am
        新文件:    homework1/makefile.in
        新文件:    homework1/missing
        新文件:    homework1/stamp-h1
        新文件:    homework1/test

meleus@meleus-laptop:~/文档/Lung good team$ 
```



Meleus HW1 Latest commit 67a496d 6 minutes ago

..		
.deps	HW1	6 minutes ago
autom4te.cache	HW1	6 minutes ago
aclocal.m4	HW1	6 minutes ago
autoscan.log	HW1	6 minutes ago
compile	HW1	6 minutes ago
config.h	HW1	6 minutes ago
config.h.in	HW1	6 minutes ago
config.log	HW1	6 minutes ago
config.status	HW1	6 minutes ago
configure	HW1	6 minutes ago
configure.ac	HW1	6 minutes ago
depcomp	HW1	6 minutes ago
funca.cpp	HW1	6 minutes ago
funca.h	HW1	6 minutes ago
funca.o	HW1	6 minutes ago

四、实验总结

1、学会了如何建立主机(虚拟机/PC，或双系统）开发环境；
2、学会了使用本地 gcc 编译应用程序；
3、掌握了如何使用 Makefile 管理应用程序，能通过 autotools 生成 Makefile，熟悉了常用的 make 操作；
4、学会了通过 git/github 管理团队软件和工作文件。