1、 解释 Linux 内核源代码的结构，以及根目录下的 Makefile 文件

```
root@Class1Server:/usr/src# ls -l
total 16
drwxr-xr-x 25 root root 4096 8月   7  2020 linux-headers-4.15.0-112
drwxr-xr-x  8 root root 4096 8月   7  2020 linux-headers-4.15.0-112-generic
drwxr-xr-x 25 root root 4096 9月  25 16:05 linux-headers-4.15.0-142
drwxr-xr-x  8 root root 4096 9月  25 16:05 linux-headers-4.15.0-142-generic
```

```
root@Class1Server: /usr/src/linux-headers-4.15.0-112
drwxr-xr-x  33 root root  4096 8月   7  2020 arch
drwxr-xr-x   3 root root  4096 8月   7  2020 block
drwxr-xr-x   2 root root  4096 8月   7  2020 certs
drwxr-xr-x   4 root root  4096 8月   7  2020 crypto
drwxr-xr-x  13 root root  4096 8月   7  2020 Documentation
drwxr-xr-x 131 root root  4096 8月   7  2020 drivers
drwxr-xr-x   2 root root  4096 8月   7  2020 firmware
drwxr-xr-x  75 root root  4096 8月   7  2020 fs
drwxr-xr-x  27 root root  4096 8月   7  2020 include
drwxr-xr-x   2 root root  4096 8月   7  2020 init
drwxr-xr-x   2 root root  4096 8月   7  2020 ipc
-rw-r--r--   1 root root  2245 1月  29  2018 Kbuild
-rw-r--r--   1 root root   287 1月  29  2018 Kconfig
drwxr-xr-x  16 root root  4096 8月   7  2020 kernel
drwxr-xr-x  14 root root  4096 8月   7  2020 lib
-rw-r--r--   1 root root 62388 7月  10  2020 Makefile
drwxr-xr-x   3 root root  4096 8月   7  2020 mm
drwxr-xr-x  69 root root  4096 8月   7  2020 net
drwxr-xr-x  28 root root  4096 8月   7  2020 samples
drwxr-xr-x  14 root root  4096 8月   7  2020 scripts
drwxr-xr-x  10 root root  4096 8月   7  2020 security
drwxr-xr-x  25 root root  4096 8月   7  2020 sound
drwxr-xr-x  29 root root  4096 8月   7  2020 tools
drwxr-xr-x   6 root root  4096 8月   7  2020 ubuntu
```

arch：这个子目录包含了此核心源代码所支持的硬件体系结构相关的核心代码。

ipc：此目录包含了核心的进程间通讯代码。

include：这个目录包括了核心的大多数 include 文件。另外对于每种支持的体系
结构分别有一个子目录。

init：此目录包含核心启动代码。

mm 此目录包含所有的内存管理代码。与具体硬件体系结构相关的内存管理代码
位于 arch/*/mm 目录下，如对应于 X86 的就是 arch/i386/mm/fault.c。

drivers：系统中所有的设备驱动都位于此目录中。它又进一步划分成几类设备驱
动，每一种也有对应的子目录，如声卡的驱动对应于 drivers/sound。

fs：Linux 支持的文件系统代码。不同的文件系统有不同的子目录对应，如 ext2 文件系统对应的就是 ext2 子目录。

kernel：主要核心代码。同时与处理器结构相关代码都放在 arch/*/kernel 目录下。

net：核心的网络部分代码。里面的每个子目录对应于网络的一个方面。

lib：此目录包含核心的库代码。与处理器结构相关库代码被放在 arch/*/lib/目录下。

scripts：此目录包含用于配置核心的脚本文件。

documentation：此目录包含一些文档，起参考作用

2、 查找 GCC 使用的库文件和头文件存放的路径

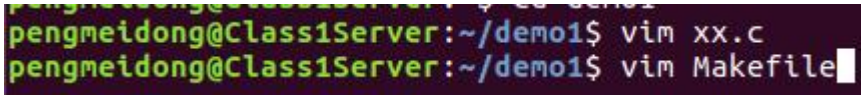
```
pengmeidong@Class1Server:/$ dpkg -S gcc
libgcc-5-dev:amd64: /usr/lib/gcc/x86_64-linux-gnu/5/include/avx512ifmavlintrin.h
linux-headers-4.15.0-112-generic: /usr/src/linux-headers-4.15.0-112-generic/incl
ude/config/have/gcc/plugins.h
linux-headers-4.15.0-112-generic: /usr/src/linux-headers-4.15.0-112-generic/scri
pts/gcc-ld
libgcc-5-dev:amd64: /usr/lib/gcc/x86_64-linux-gnu/5/include/avx512ifmaintrin.h
linux-headers-4.15.0-142: /usr/src/linux-headers-4.15.0-142/include/dt-bindings/
reset/qcom,gcc-msm8960.h
libgcc-5-dev:amd64: /usr/lib/gcc/x86_64-linux-gnu/5/include/backtrace-supported.
h
linux-headers-4.15.0-112: /usr/src/linux-headers-4.15.0-112/include/acpi/platfor
m/acgccex.h
libgcc-5-dev:amd64: /usr/lib/gcc/x86_64-linux-gnu/5/libasan.so
libgcc-5-dev:amd64: /usr/lib/gcc/x86_64-linux-gnu/5/include/avx512cdintrin.h
libgcc-5-dev:amd64: /usr/lib/gcc/x86_64-linux-gnu/5/libitm.so
libgcc-5-dev:amd64: /usr/lib/gcc/x86_64-linux-gnu/5/include/clwbintrin.h
gcc-5: /usr/bin/x86_64-linux-gnu-gcc-ranlib-5
gcc-5: /usr/share/doc/gcc-5-base/sanitizer
linux-headers-4.15.0-142: /usr/src/linux-headers-4.15.0-142/include/linux/libgcc
.h
libgcc1:amd64: /usr/share/lintian/overrides/libgcc1
libgcc-5-dev:amd64: /usr/lib/gcc/x86_64-linux-gnu/5/include/tmmintrin.h
libstdc++-5-dev:amd64, g++-5: /usr/share/doc/gcc-5-base/C++
```

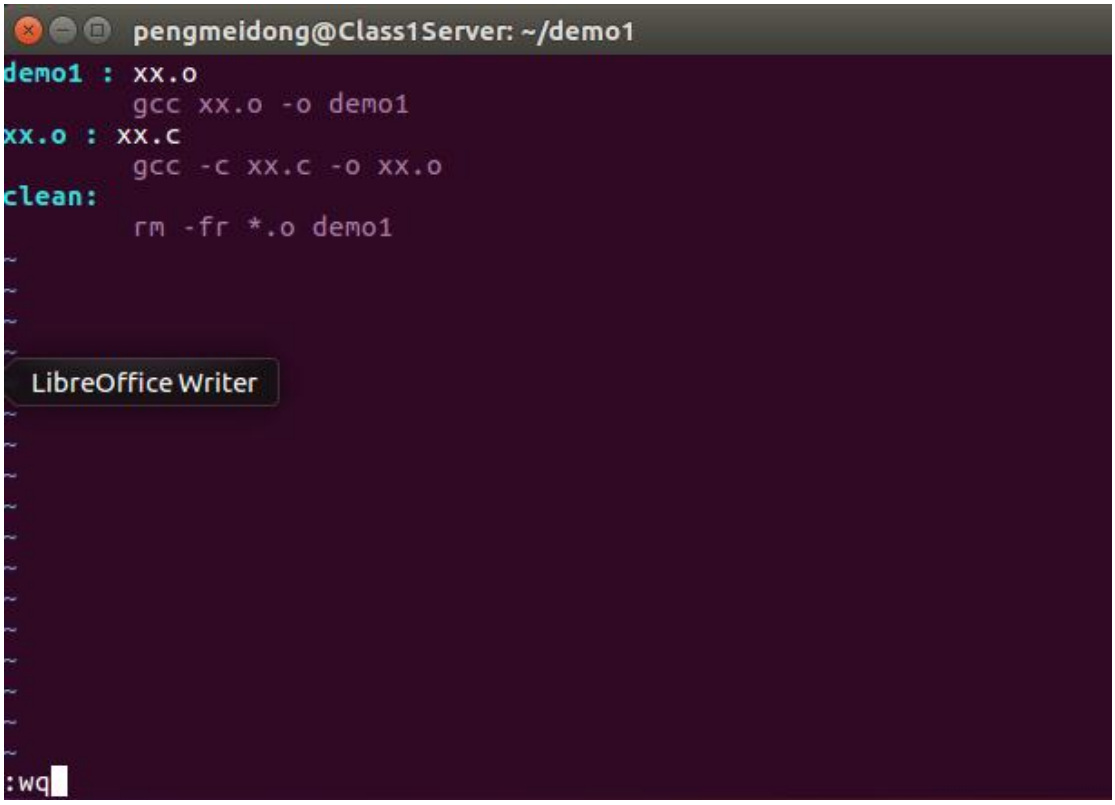
其中的.h 文件即为头文件，.so 等文件即为库文件，.gz 文件则为压缩包文件

3、 编写一个 C 程序，其功能是用户向系统输入一个简单的问候，然后系统以字母反序将该问候显示出来

（1）新建文件夹 demo1，在其中新建 xx.c 文件

（2）新建 Makefile 文件



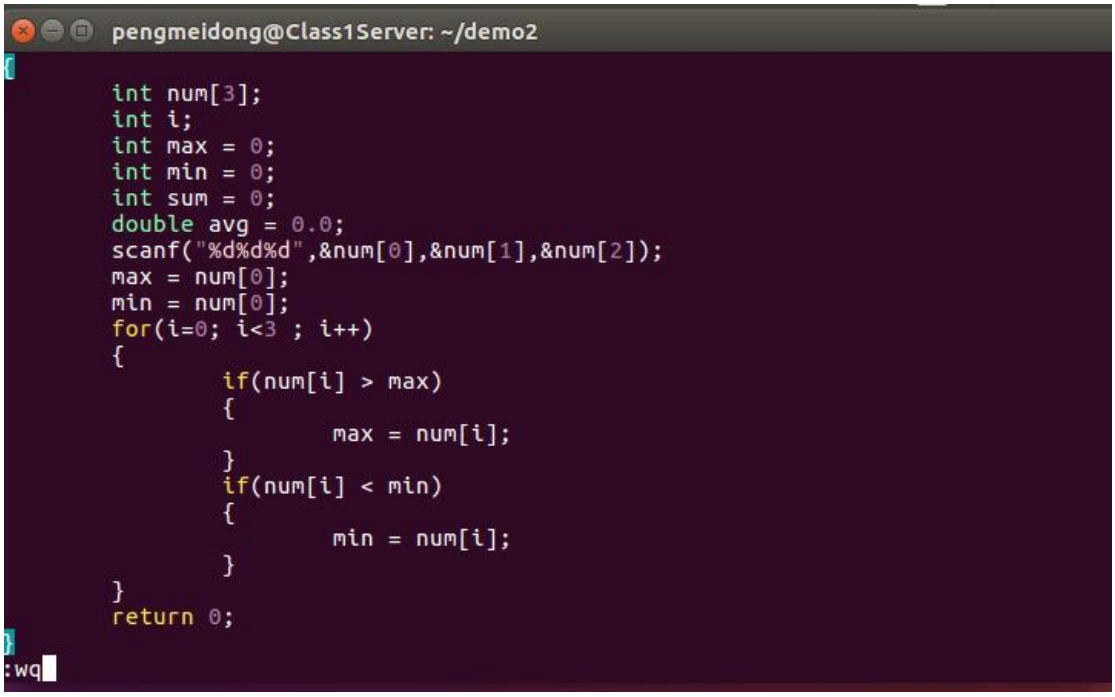


（3）编译

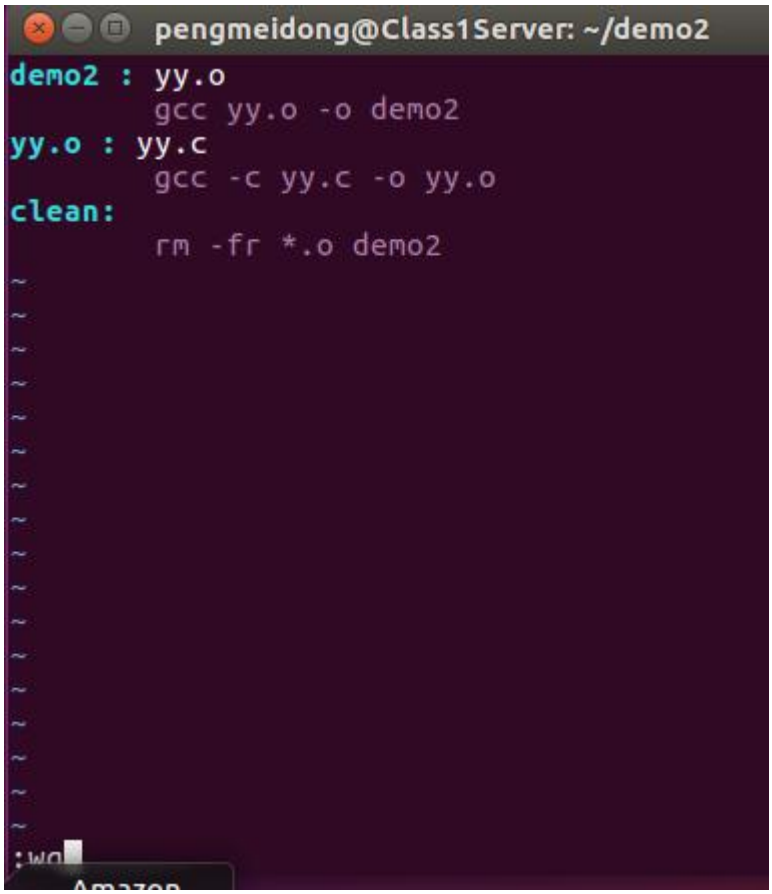
```
pengmeidong@Class1Server:~/demo1$ make
gcc -c xx.c -o xx.o
xx.c: In function ‘main’:
xx.c:7:2: warning: implicit declaration of function ‘gets’ [-Wimplicit-function-
declaration]
  gets(str);
  ^
gcc xx.o -o demo1
xx.o: In function `main':
xx.c:(.text+0x37): warning: the `gets' function is dangerous and should not be u
sed.
```

4、 编写一个 C 程序，其功能是从键盘输入三个整数，然后系统输出最大值、最小值和平均值（保留两位小数）。要求：使用函数功能及 Makefile 文件

（1）新建 demo2 文件夹，新建 yy.c 文件



（2）新建 Makefile 文件



（3）编译

```
pengmeidong@Class1Server:~/demo2$ make
gcc -c yy.c -o yy.o
gcc yy.o -o demo2
```

```
pengmeidong@Class1Server:~/demo2$ ./demo2
23
34
12
the max is 34
the min is 12
the average is 23.000000
```

5. 编写一个多个文件的 c 程序，其功能是从键盘输入一个字符串 name，然后系统输出 Hello name[]

（1）新建 mypro 文件夹，进入该文件夹

```
pengmeidong@Class1Server:~$ mkdir mypro
pengmeidong@Class1Server:~$ cd mypro
pengmeidong@Class1Server:~/mypro$
```

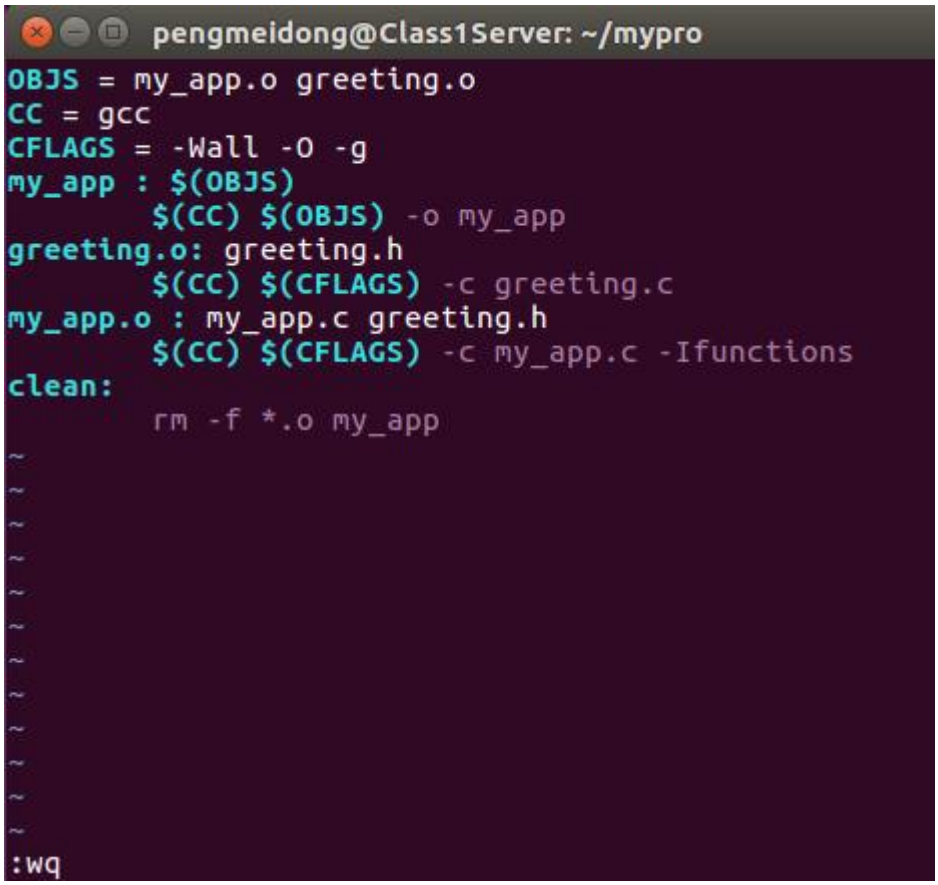
（2）新建 greeting.h 文件

```
pengmeidong@Class1Server: ~/mypro
#ifndef GREETING_H_INCLUDED
#define GREETING_H_INCLUDED
void greeting ( char * name );
#endif
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
:wq
```

（3）新建 greeting.c 文件

（4）新建 my_app.c 文件

（5）新建 Makefile 文件



（6）编译，测试

```
pengmeidong@Class1Server:~/mypro$ vim greeting.h
pengmeidong@Class1Server:~/mypro$ vim greeting.c
pengmeidong@Class1Server:~/mypro$ vim my_app.c
pengmeidong@Class1Server:~/mypro$ vim Makefile
pengmeidong@Class1Server:~/mypro$ make
gcc -Wall -O -g -c my_app.c -Ifunctions
my_app.c: In function ‘main’:
my_app.c:7:2: warning: ignoring return value of ‘scanf’, declared with attribute
 warn_unused_result [-Wunused-result]
  scanf("%s",name);
  ^
gcc -Wall -O -g -c greeting.c
gcc my_app.o greeting.o -o my_app
pengmeidong@Class1Server:~/mypro$ ./my_app
Your name please:pengmeidong
Hello pengmeidong!
pengmeidong@Class1Server:~/mypro$
```