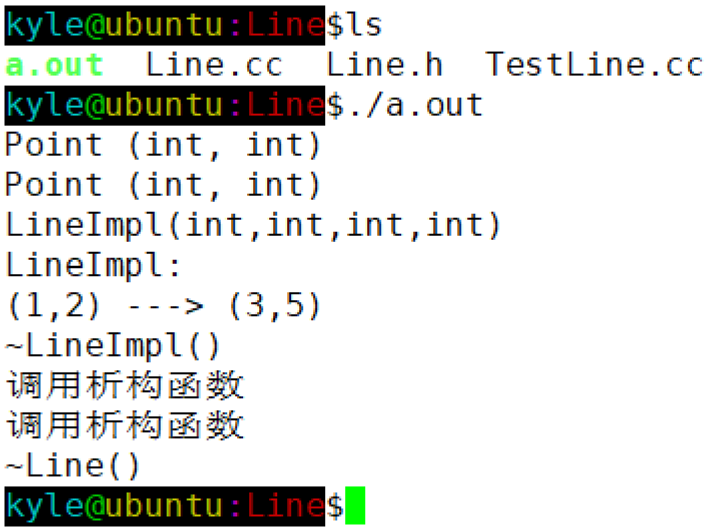
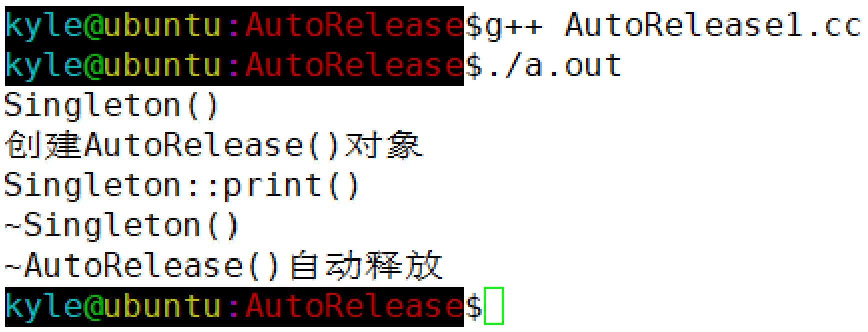
20190115

1. 实现Line类的PIMPL模式

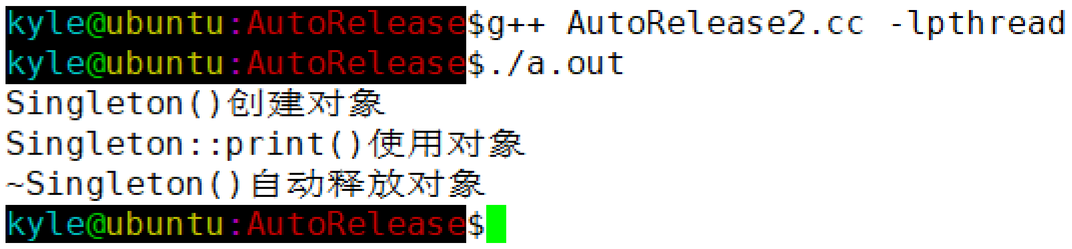


```
kyle@ubuntu:Line$ls
a.out  Line.cc  Line.h  TestLine.cc
kyle@ubuntu:Line$./a.out
Point (int, int)
Point (int, int)
LineImpl(int,int,int,int)
LineImpl:
(1,2) ---> (3,5)
~LineImpl()
调用析构函数
调用析构函数
~Line()
kyle@ubuntu:Line$
```

2. 实现单例模式的自动释放（3种方式）

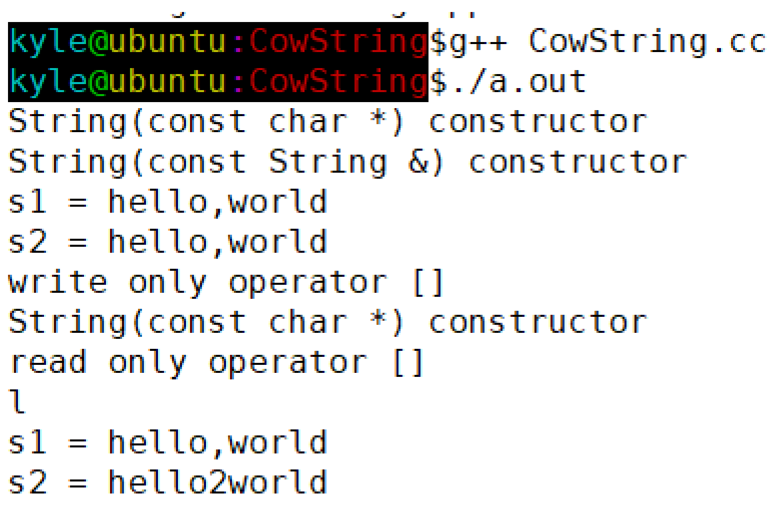


```
kyle@ubuntu:AutoRelease$g++ AutoRelease1.cc
kyle@ubuntu:AutoRelease$./a.out
Singleton()
创建AutoRelease()对象
Singleton::print()
~Singleton()
~AutoRelease()自动释放
kyle@ubuntu:AutoRelease$
```



```
kyle@ubuntu:AutoRelease$g++ AutoRelease2.cc -lpthread
kyle@ubuntu:AutoRelease$./a.out
Singleton()创建对象
Singleton::print()使用对象
~Singleton()自动释放对象
kyle@ubuntu:AutoRelease$
```

3. 实现COW的String，让其operator[]能够区分出读写操作



```
kyle@ubuntu:CowString$g++ CowString.cc
kyle@ubuntu:CowString$./a.out
String(const char *) constructor
String(const String &) constructor
s1 = hello,world
s2 = hello,world
write only operator []
String(const char *) constructor
read only operator []
l
s1 = hello,world
s2 = hello2world
```