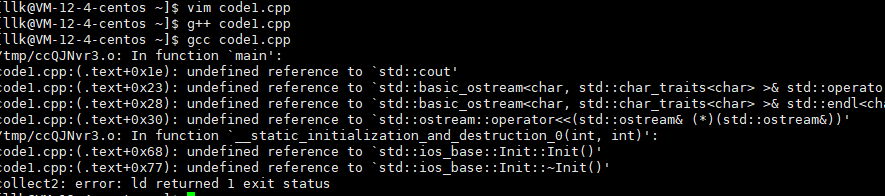
//开发的工具gcc/g++，但是gcc只能编译c，g++能够编译c和c++

##### 编译方面需要注意的

###### 使用gcc编译程序

1.使用gcc编译c++代码会出现错误，因为无法进行识别



2.使用指令 gcc xxx.c 就能对这个源码进行编译，编译后生成默认的 a.out ，使用指令 ./a.out 就能进行运行程序了

3.从源文件编译成系统可识别的二进制可执行文件需要下面的四个步骤  
（1）预处理：去注释、头文件展开、条件编译、宏替换

//头文件展开其实就是将头文件中的内容拷贝到那个.i的中间文件中

//语法检查、语义检查之类的是在预处理阶段之前进行的

//具体的什么阶段生成对应的中间文件，参考c的笔记

//预处理之后还是c语言

（2）编译：生成汇编 -

//编译过后形成汇编语言，汇编语言还是能够看的懂的

（3）汇编：

//汇编形成的是机器可识别代码

（4）连接

//连接形成的是可执行文件或者库文件

4.为什么能够在win或者Linux上进行c/c++或者其他形式的开发呢？

（1）我们的系统一定要有c/c++开发相关的头文件库文件等文件

（2）在Linux上这些头文件库文件存放的位置，，默认是 /usr/include 这个路径下。比如/usr/include/stdio.h

（3）我们在安装vs2019的时候，我们需要选择对应的开发包，同步也在下载c的头文件和库文件。这个头文件和库文件并不是系统自带的，而是下载的

//使用头文件的时候，#include<stdio.h> 而不是使用#include</usr/bin/stdio.h> 原因就是因为下载软件的时候这些库是软件帮你安装的,因此使用的时候不用再去输入具体的路径

//在某些平台下面，所谓的配置环境变量，就是将下载的相应的库函数和某些包进行下载。以及在设置中表明这些文件的路径

###### 2. gcc的分支选项 gcc-E

1.这个选项可以单独运行预处理阶段，并且可以搭配上 -o选项，将预处理后的内容指定的输出到指定的文件上面

//使用格式为: gcc -E test.c -o xxx.c 或者gcc -E test.c -o xxx.i -----前者可以将编译后的内容指定的输入到某个文件，如果文件不存在会自动创建出来这个文件。

###### 3.在命令行中定义宏 gcc-D

1.使用的格式为 gcc test.c -Dxxx xxxx为宏的名字，此外这个宏还能在命令行中定义值，也就是gcc test.c -Dxxx=5

2.使用这个命令一般情况下用于条件编译

//至于为什么能在外部定义宏，毕竟Linux天然的就能修改里面的文本

###### 4.补充的题外话 社区版和专业版

1.社区版就是专业版的阉割版本，这就是条件编译的应用场景

##### 二、编译文件细节

###### 1. gcc -o 选项

1.如果gcc没有选项，那么就是直接生成可执行文件，但是生成的文件的名称基本上都是默认的a.out

//所以推荐的编译写法是这样 gcc test.c -o xxxx 就能生成指定的名字的可执行文件了

//这样也能写 gcc -o xxxx test.c 这样也能自定义可执行文件的名字

//使用这个 -o 选项，那么在-o后面跟的必须是自定义的名字，而不是源文件

###### 2. gcc -E 选项 -----进行预处理的指令

//-E 选项就是只进行预处理这个功能

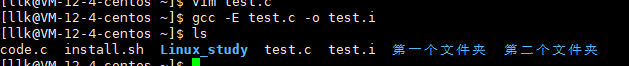
1.使用格式为 gcc -E test.c 如果就这样结束，会将预处理后的内容都打印在屏幕上面。

//- E选项后面跟随的是源代码文件

//- E选项是只对源文件进行预处理

2.推荐和 -o 搭配使用，并且 -o 后面跟随的文件名要以 .i 为后缀名

//也就是只进行预处理后的文件是以 .i 为后缀的 gcc -E test.c -o xxx.i 就是常用的使用格式



###### 3.gcc -S 选项 -----进行编译的指令

// -S 选项是只进行到编译这个过程

//编译后形成的是汇编语言

1.常用的使用格式还是和 -o搭配使用 gcc -S test.i -o test.s 编译后的文件是以 .s 结尾的



2.如果-S后面跟随的还是.c的源文件，那么还是会只进行到编译这一步停止。

//使用的格式依然是 gcc -S test.c -o test.s

//一定要注意编译后形成的文件，里面是汇编代码

###### 4. gcc -c ----进行汇编的指令

// -c 选项是只进行到汇编这个过程

//汇编后形成的是二进制的机器指令

1.使用的格式 gcc -c test.s -o test.o 汇编后的文件是以 .o 结尾的



2.当然后面 -c 选项后面还是能跟上 .c 的源文件，还是只进行到汇编这个过程停止

3.这个 .o 文件叫做可重定位目标二进制文件，简称目标文件。 在win的平台下这个文件是以 .obj 结尾的

4.这个文件并不能够独立执行，需要经过 链接 才能进行执行。 给可执行权限与这个文件能不能执行并没有必然的关系

5.这个文件也能经过gcc进行编译 gcc test.o -o test 这样就行成了一个可执行文件

###### 5.链接的过程

1. 链接就是将可重定位目标二进制文件，和库进行链接形成可执行程序

2.库是我们提供方法的实现。 就像c语言的标准库，本质上就是一个文件，有路径

3.在Linux上 ：.so结尾的叫动态库 .a结尾的叫静态库

4.在win上： .dll结尾的叫动态库 .lib结尾的叫静态库

5.库都是有自己的命名规则的： 比如 libname.so.xxx lib前缀 .so后缀 xxx：版本号之类的，有的可能不显示 name：库的名字

//lib只是代表是一个库，后面的.so才是代表的静态库 并且这个是Linux上面的命名规则

//并且在我们的机器上面，默认只会安装动态库，静态库是默认没有安装的

//方法的实现就在库当中，库其实就是把源文件（.c），经过一定的翻译，然后打包----只给你提供一个文件，不用太多的源文件

6.头文件提供方法的声明，库文件提供方法的实现+你的代码=你的软件

7.编译型语言，安装开发包，必定是下载安装对应的头文件+库文件

###### 6. .o 文件和库是如何链接的

1.动态链接

//用一个简单得例子就是，你列出了今天的十项计划安排，其余九项都是在家里面完成的，只有一件事需要去最近的警察局办理身份证，办理完身份证继续办其他事。去警察局的路径是出门往东800米。这个路径是你的邻居告诉你的。

//上面的警察局就相当于一个动态库，因为办理身份证的这个办法只能在警察局里实现，也就是跳转到库中执行（在代码中对应着指向相应的方法），办理完身份证回家就相当于返回原先的代码调用处。你的这个邻居就是编译器。

//一旦这个派出所搬迁那么附近的居民就无法再班里身份证

（1）动态库一般也被称为共享库。附近的居民办理身份证只需要到这个派出所就可以，所以只需要一个就可以

//动态库不能缺失，一旦对应的动态库缺失，影响的可能就是很多的程序

2.静态链接

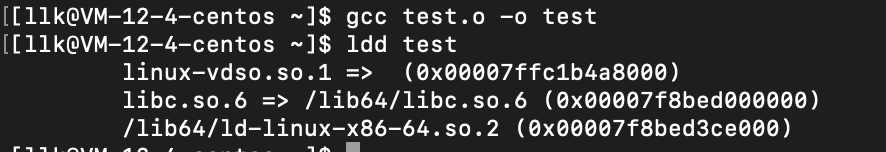
//再根据上面的小故事，派出所入户办理身份证。

（1）户籍警直接上门帮你办理，就算派出所搬迁了，也不影响你办理身份证。 此时的派出所就是静态库，在哪都行

//在编译器使用静态库进行静态链接的时候，会将自己的方法拷贝到目标程序中，就算该静态库删除，也不会影响该程序的正常运行

##### 三、查看可执行软件链接的动态库

1.使用命令 ldd ，具体使用的格式为 ldd xxxxx xxxx为可执行文件



2.其实很多Linux上面的指令都是c/c++ 实现的，所以也可以用ldd查看这些指令链接地动态库



//当前路径下面没有这个可执行文件的，需要在ldd后面跟上详细的可执行文件的路径

##### 四、使用静态库方面的一些问题

1.在Linux上，编译形成的可执行文件，默认采用的就是动态链接

2.动态库只能提供动态链接 ，静态库只能提供静态链接

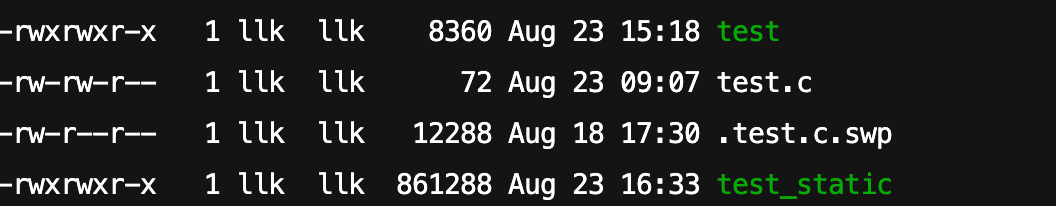
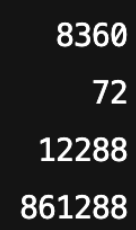
###### 1.使用静态库的指令 -static

1.使用静态库形成的可执行文件还是同样可以运行的，只是链接的时候使用的静态库

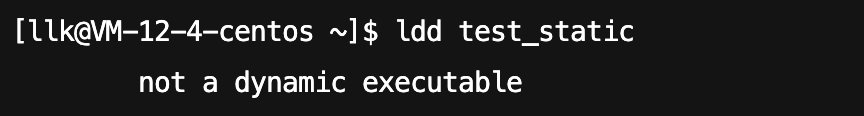
//因为链接的时候默认使用的是动态库，所以使用静态库的时候需要指令来调用，指令格式为 gcc test.c -o test\_static -static

//但是使用静态链接之后，生成的执行文件体积会比相同源文件生成的可执行文件内存更大

也就是说平时我们很少使用静态链接

 8360 861288

2.使用静态链接也就无法使用ldd查看链接的库



###### 2.静态库的安装

1.想要使用静态的链接，必须下载静态库，但是一般情况下，gcc个g++上面是没有安装静态库的

2.在网上直接搜索对应的系统进行安装： 例如： centos7 yum安装c/c++的静态库

//sudo yum install glibc-static sudo yum install libstdc++-static

第一个是安装的c的静态库，第二个安装的事c++的静态库

###### 3.没有静态库的情况下

1.没有静态库的情况下我们就不能使用-static

2.但是如果我们没有动态库，只有静态库，编译的时候是能通过的。

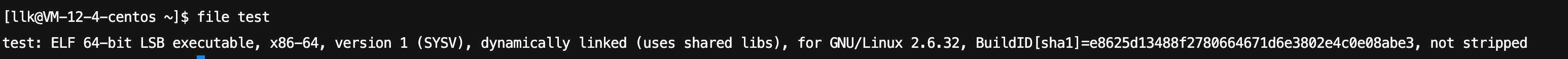
//因为Linux默认使用的是动态库，没有动态库会自动链接静态库。如果两个都没有就会报错

//也就是说-static的本质就是改变优先级

3.所以我们的可执行程序不一定是纯的静态链接或者动态链接

##### 五、file指令

1.file指令能够对可执行程序进行查看是动态链接还是静态链接



//除此之外还会包括很多的信息，比如拓展名啊，什么的

##### 六、动静态的优缺点

###### 1.动态库

1.优点

（1）动态库因为是共享库，可以有效的节省资源（磁盘空间、内存空间、网络空间）

2.缺点

（1）动态库一旦缺失，导致使用这个库的程序都会无法运行

###### 2.静态库

1.优点

（1）静态库不依赖库，程序可以独立的运行，在同平台上面的移植性更好

2.缺点

（1）体积大，比较消耗资源

##### 七、debug 和 release

//平时我们自己在测试的时候，用的都是debug模式，在交给测试人员的时候，一般提交的都是release模式。并且用户进行使用的也是release下的程序

//我们在Linux下面生成的可执行程序，默认是release版本的

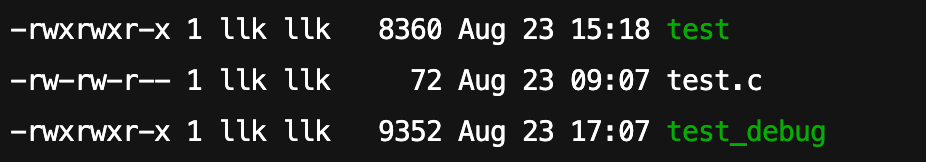
###### 1.debug

1.debug 可以被追踪，可以被调试 -----形成可执行程序的时候，添加了debug信息

//也就是说这个可执行程序的体积较大，因为包含了很多信息

2.生成debug版本的指令 -g 选项

//不管是以静态链接生成的，还是以动态链接生成的，debug版本下的内存就是比release版本下的内存大



3.拓展： 可执行程序形成的时候，不是无序的二进制构成，有自己的格式 ----可执行程序有自己的二进制格式 ELF格式

《程序员的自我修养》学完Linux《深入理解计算机系统》

4.上面的所有知识点也同样适用c++程序的编写

##### 八、Make和Makeile 自动化构建工具

1.make 是一条指令

2.makefile 是一个当前目录下的文件