动态编译

什么叫动态编译？

动态编译是将应用程序需要的模块都编译成动态链接库，启动程序（初始化）时，这些模块不会被加载，运行时用到哪个模块就调用哪个。动态编译将`.c`文件编译成`.o`或`.a`或`.lib`格式的动态链接库。

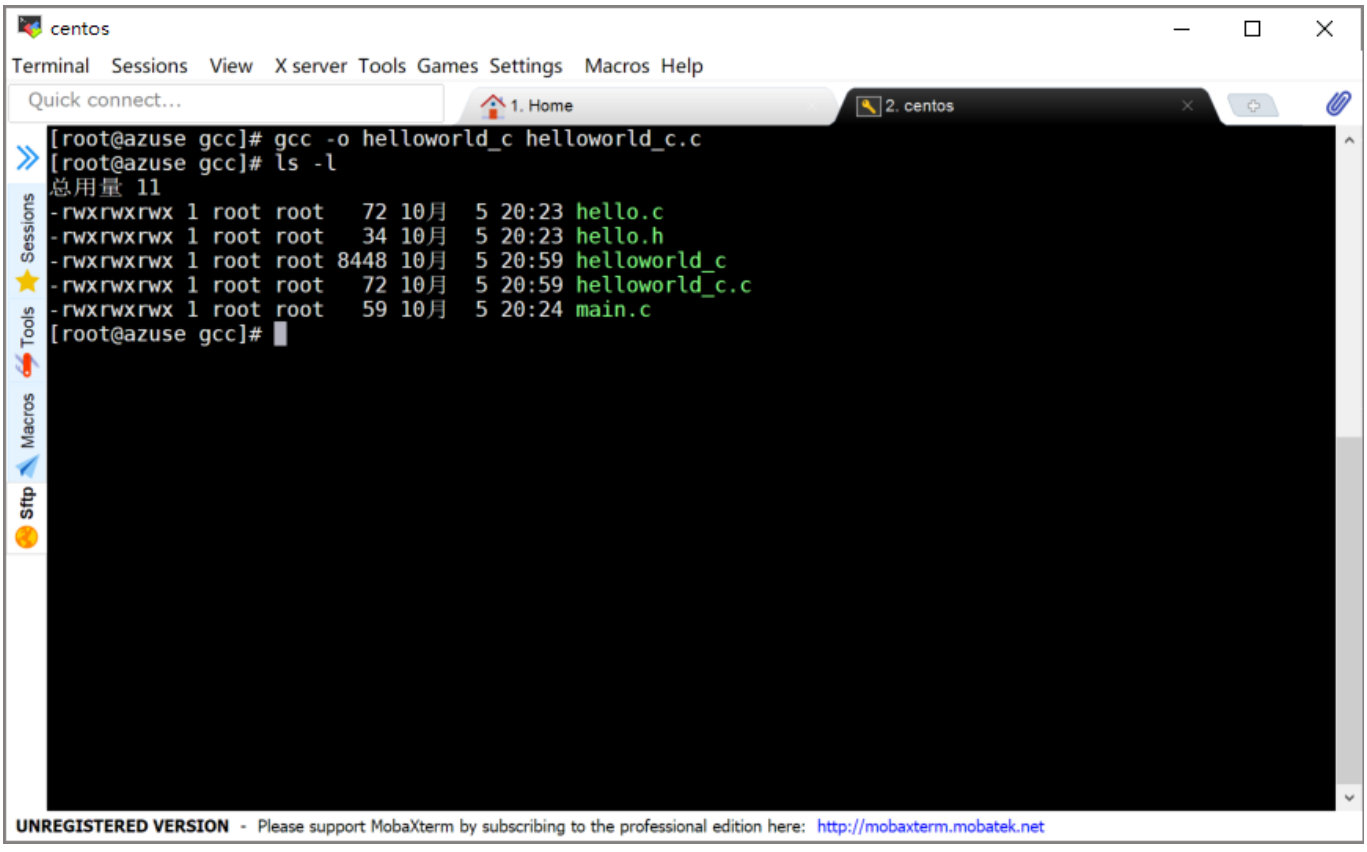
`.o .obj`： object files, they contain the output of the compiler generated code. It is still in an intermediate format, for example, most references are still unresolved. Usually there is a one to one mapping between the source file and the object file.

`.a .lib` : archives, also known as library, and are a set of object files.

一般动态编译的程序调用的系统函数默认位于系统的共享库中，而不在可执行文件中，gcc与g++默认都采用这种方式，无需外加参数。

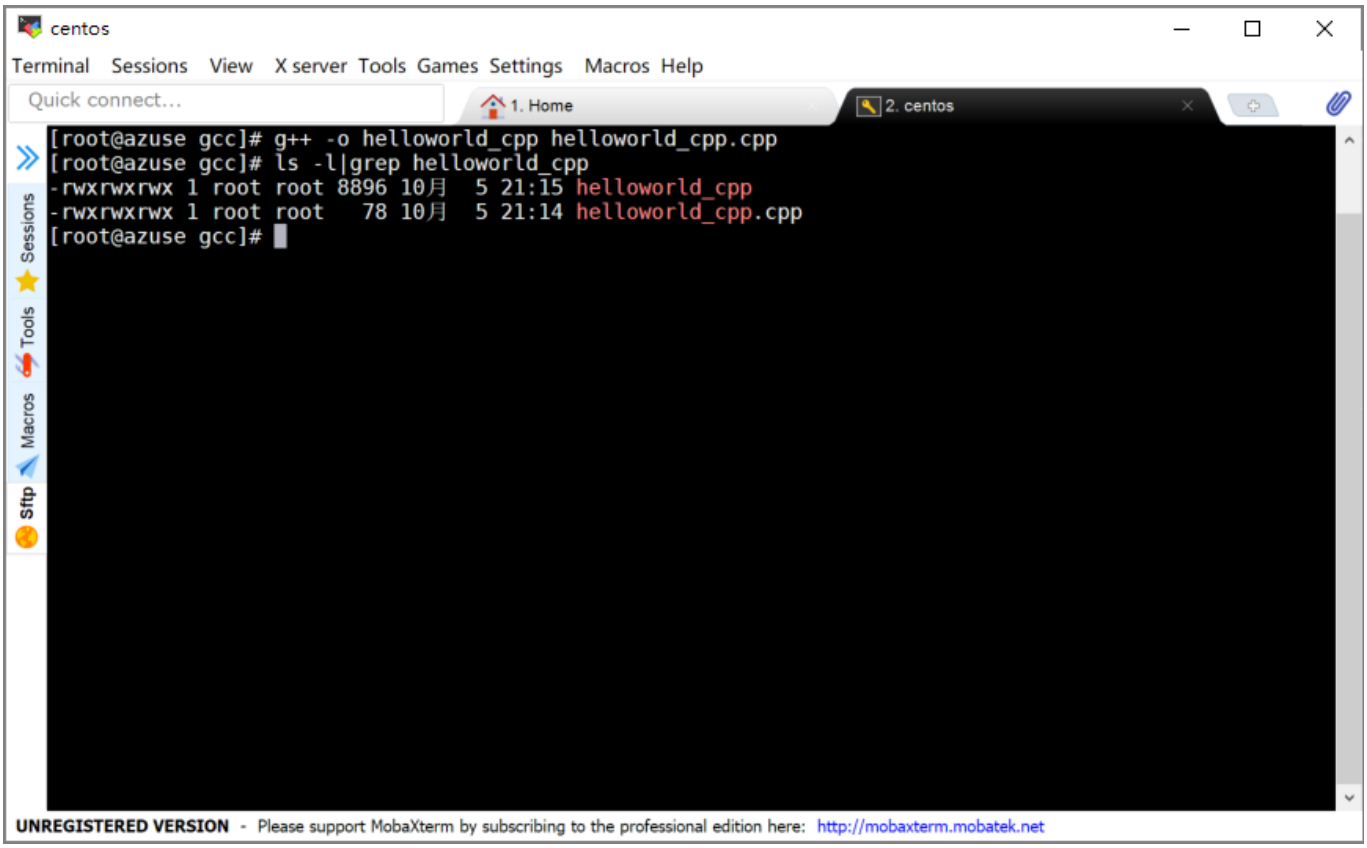
动态编译`printf("hello world");`

`gcc -o helloworld_c helloworld_c.c`
生成的helloworld_c大小为8448字节



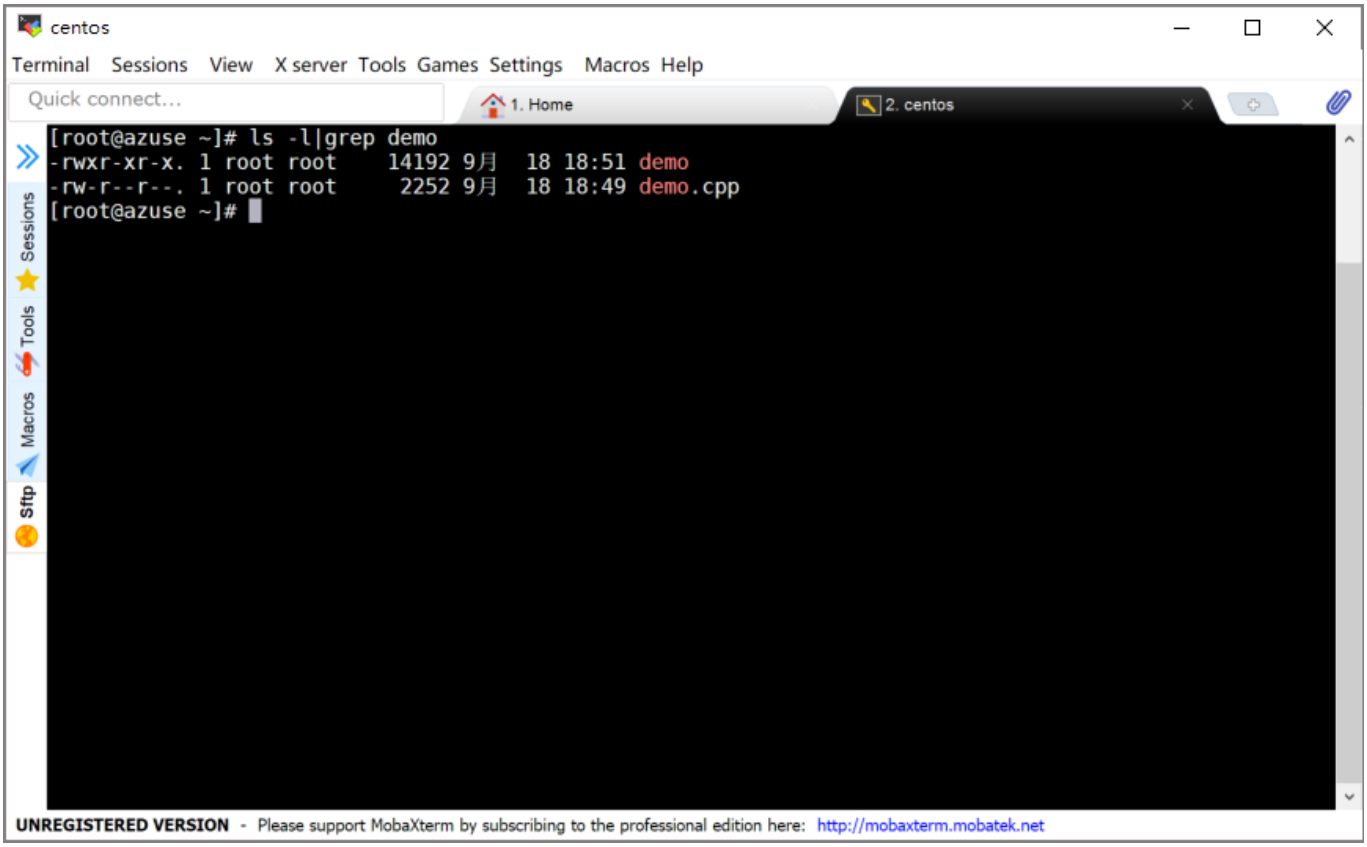
动态编译`cout << "hello, world";`

`g++ -o helloworld_cpp helloworld_cpp.cpp`
生成的helloworld_cpp大小为8896字节



第一周给出的mysql_demo.cpp的动态编译命令的可执行文件字节数

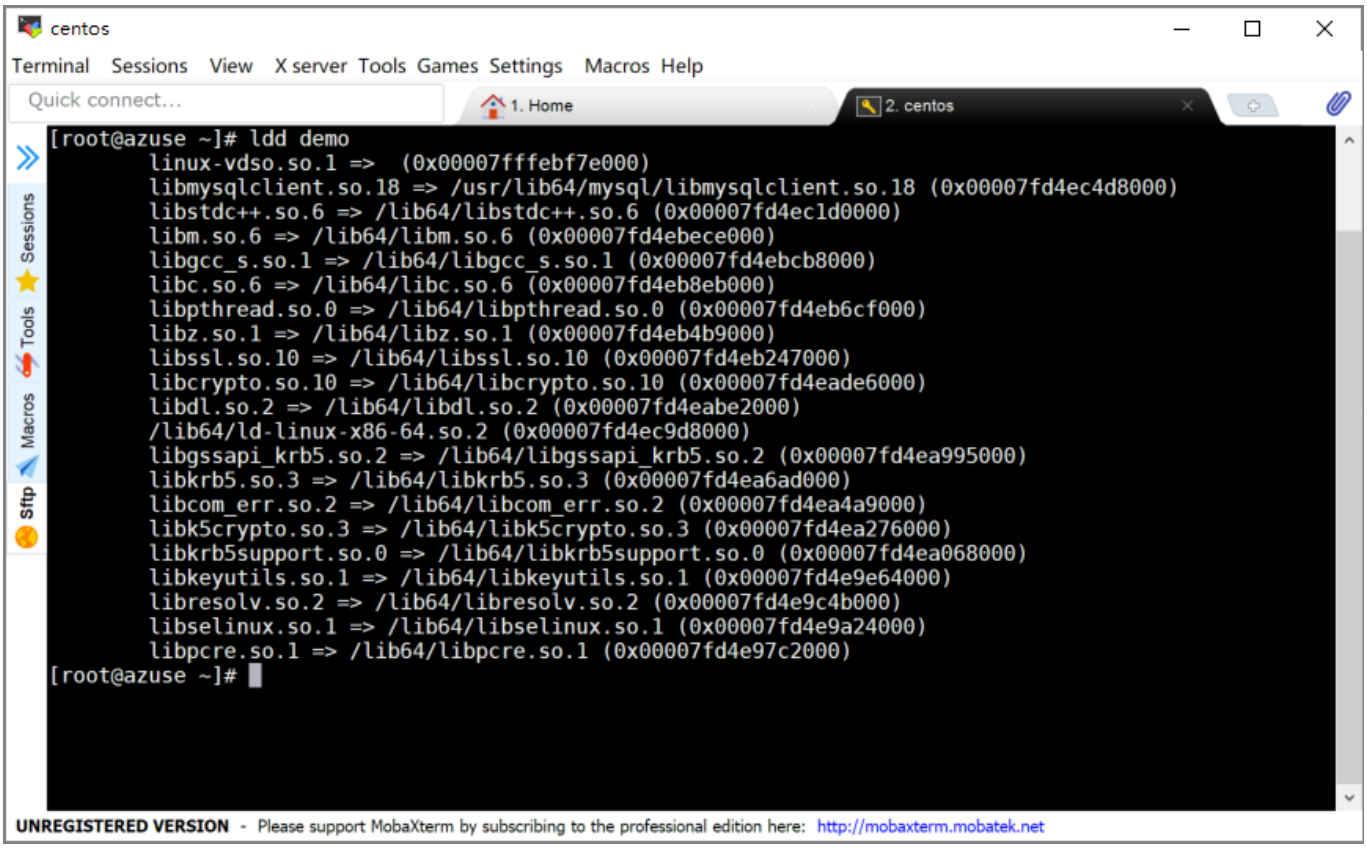
生成的demo文件大小为14192字节



如何查找某个可执行文件的动态链接库？

使用ldd命令

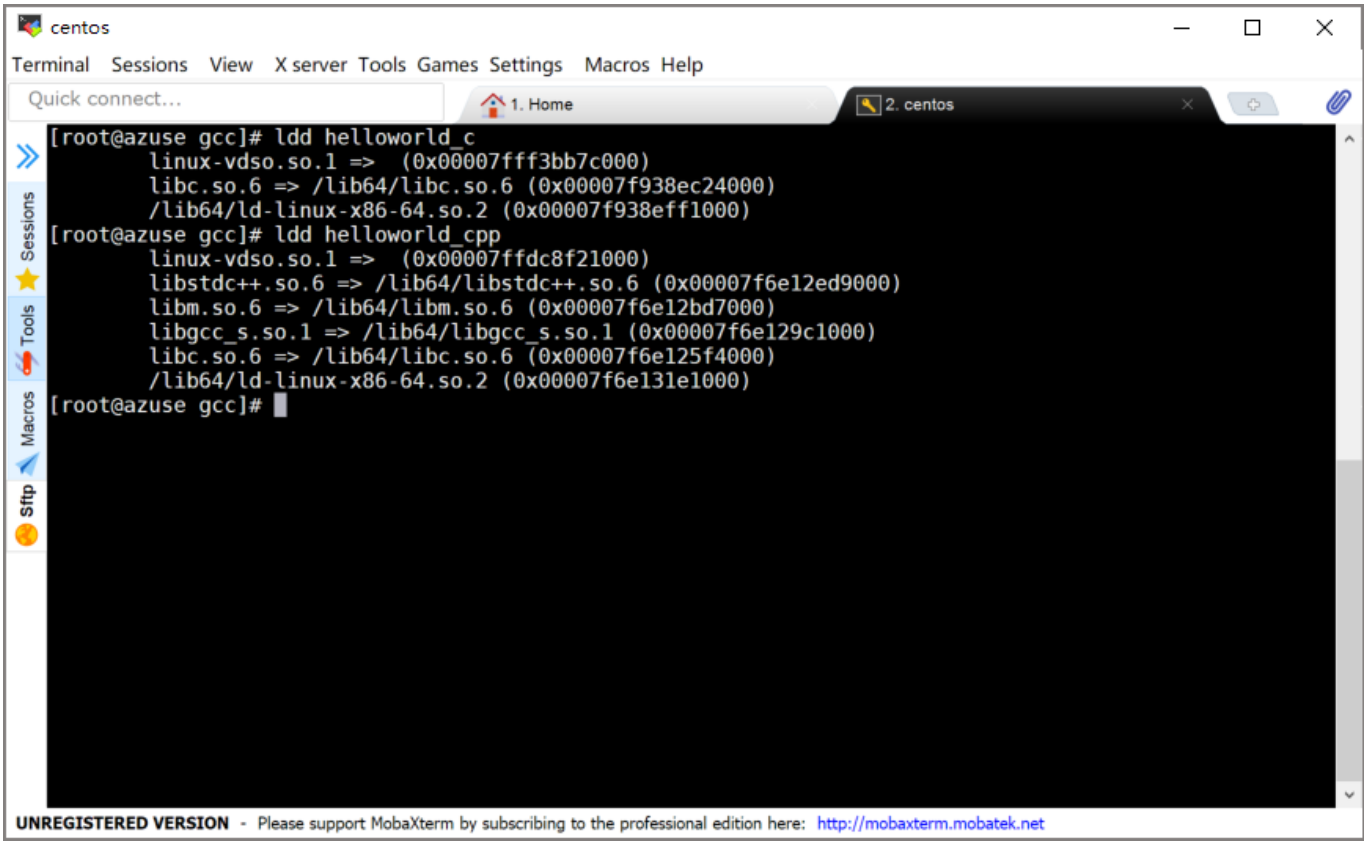
demo文件的动态链接库ldd demo：



helloworld_c和helloworld_cpp使用的动态链接库：

ldd helloworld_c

ldd helloworld_cpp



静态编译

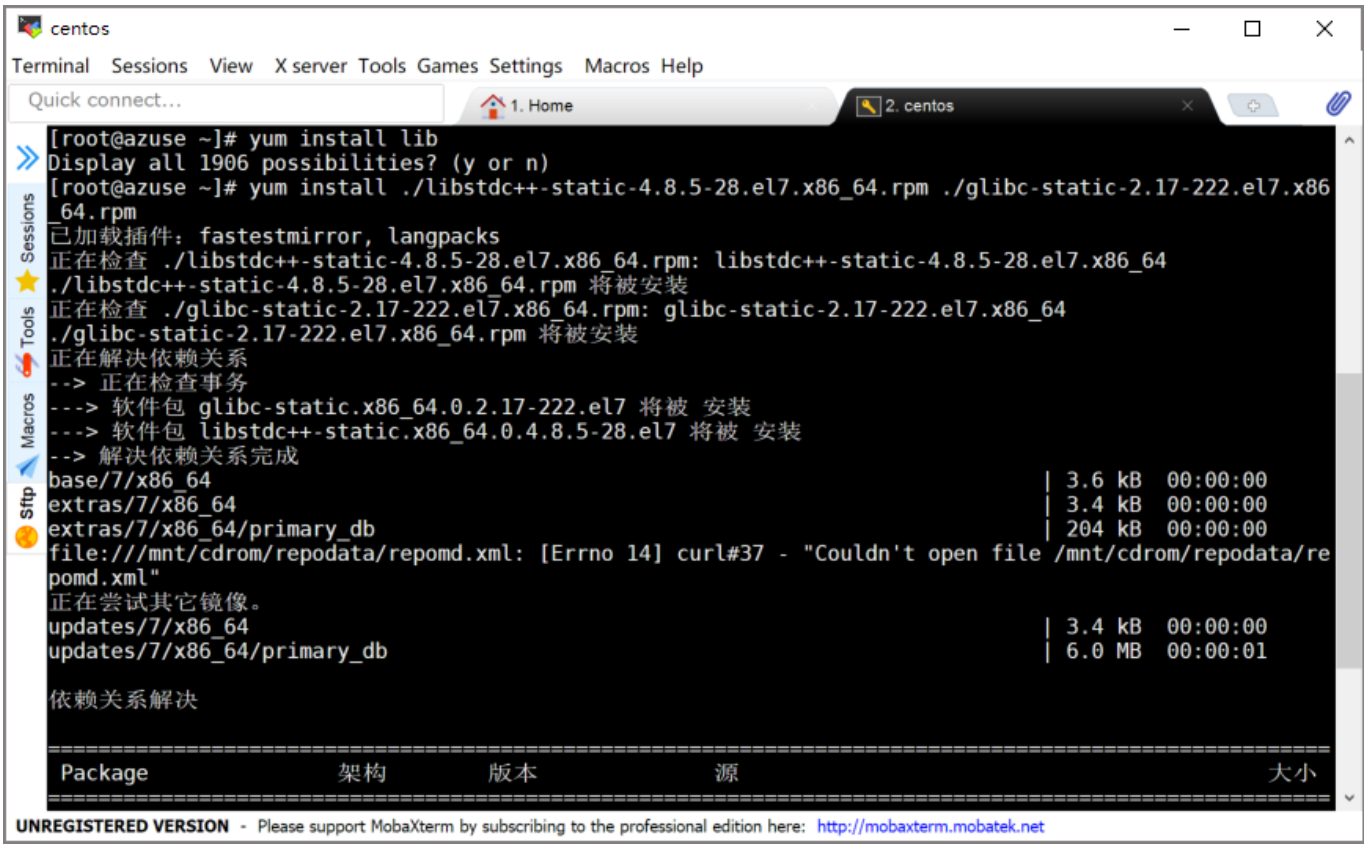
什么叫静态编译？

静态编译就是在编译时，把所有模块都编译进可执行文件里，当启动这个可执行文件时，所有模块都被加载进来。这样产生的可执行文件显然更浪费空间，但是优点是他们可以在不依赖于系统的共享库也能正常运行

静态编译`printf("hello, world");`

需要先安装libstdc++-static与glibc-static
将下载的rpm包放到/root下，并使用yum安装

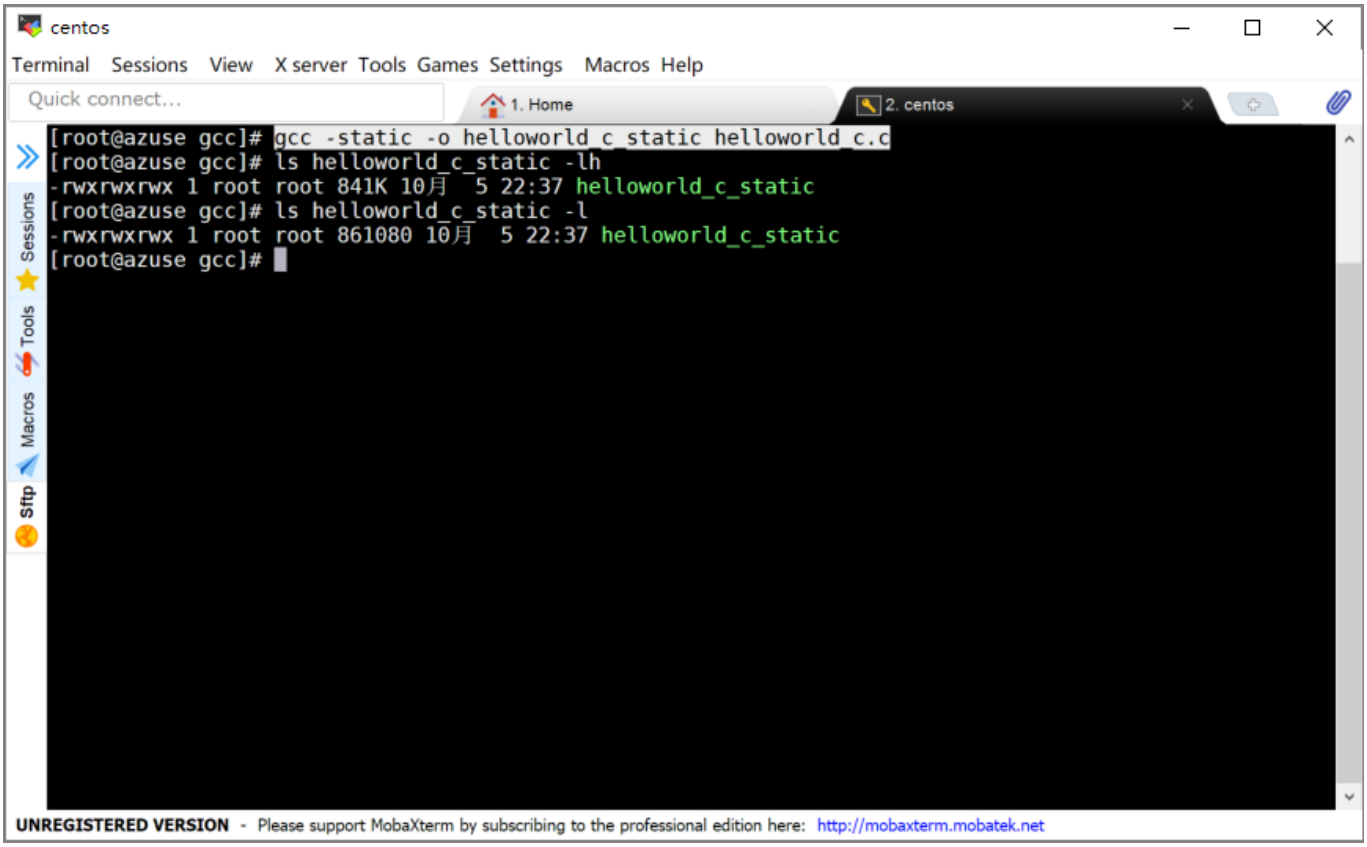
```
yum install ./libstdc++-static-4.8.5-28.el7.x86_64.rpm ./glibc-static-2.17-222.el7.x86_64.rpm
```



静态编译命令：

```
gcc -static -o helloworld_c_static helloworld_c.c
```

生成的helloworld_c_static文件大小为841KB

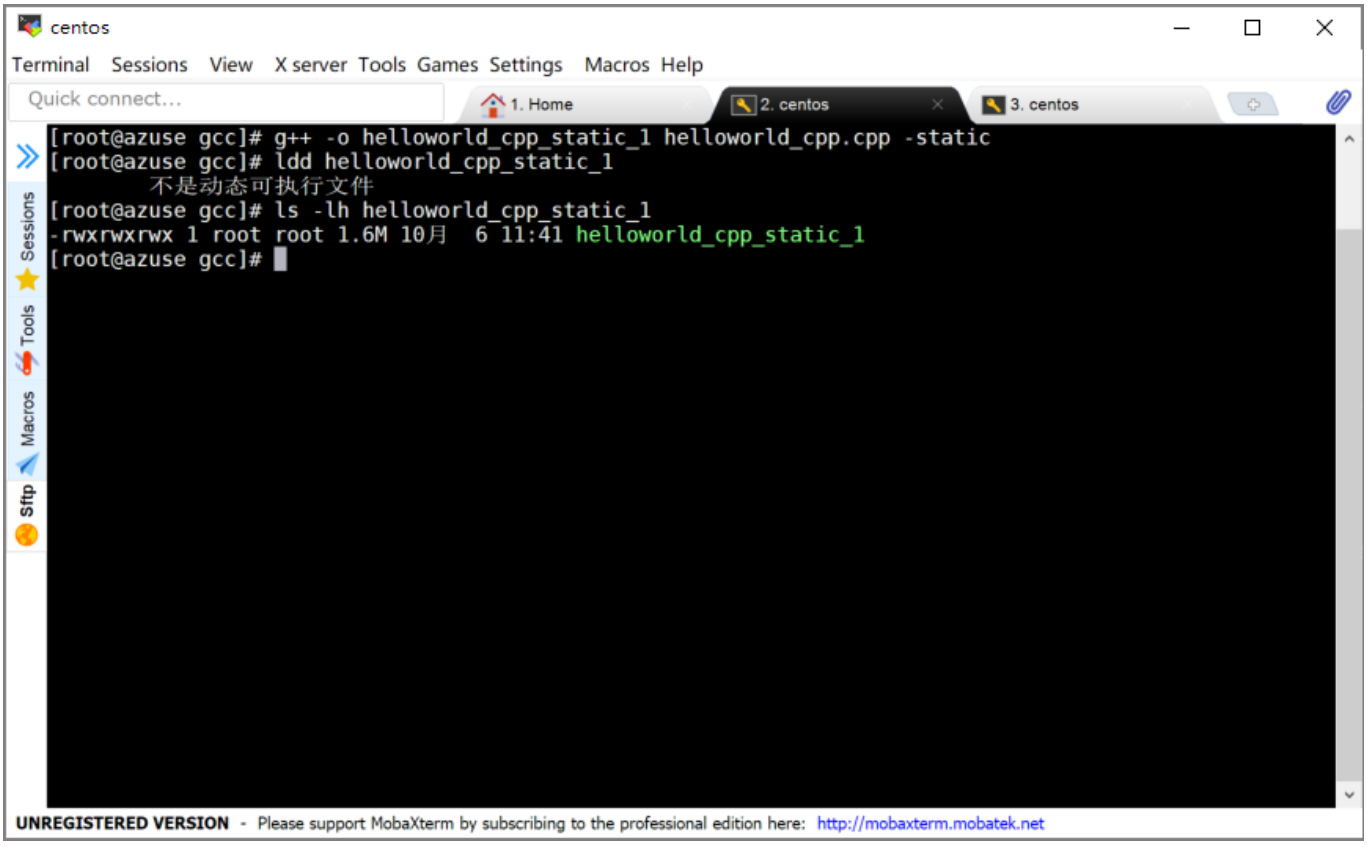


静态编译`cout << "hello, world";`

需要libstc++-static静态c++库和glibc-static，生成helloworld_cpp_staic_1

```
g++ -o helleworld_cpp_static_1 helloworld_cpp.cpp -static
```

是完全静态的文件，大小为1.6MB



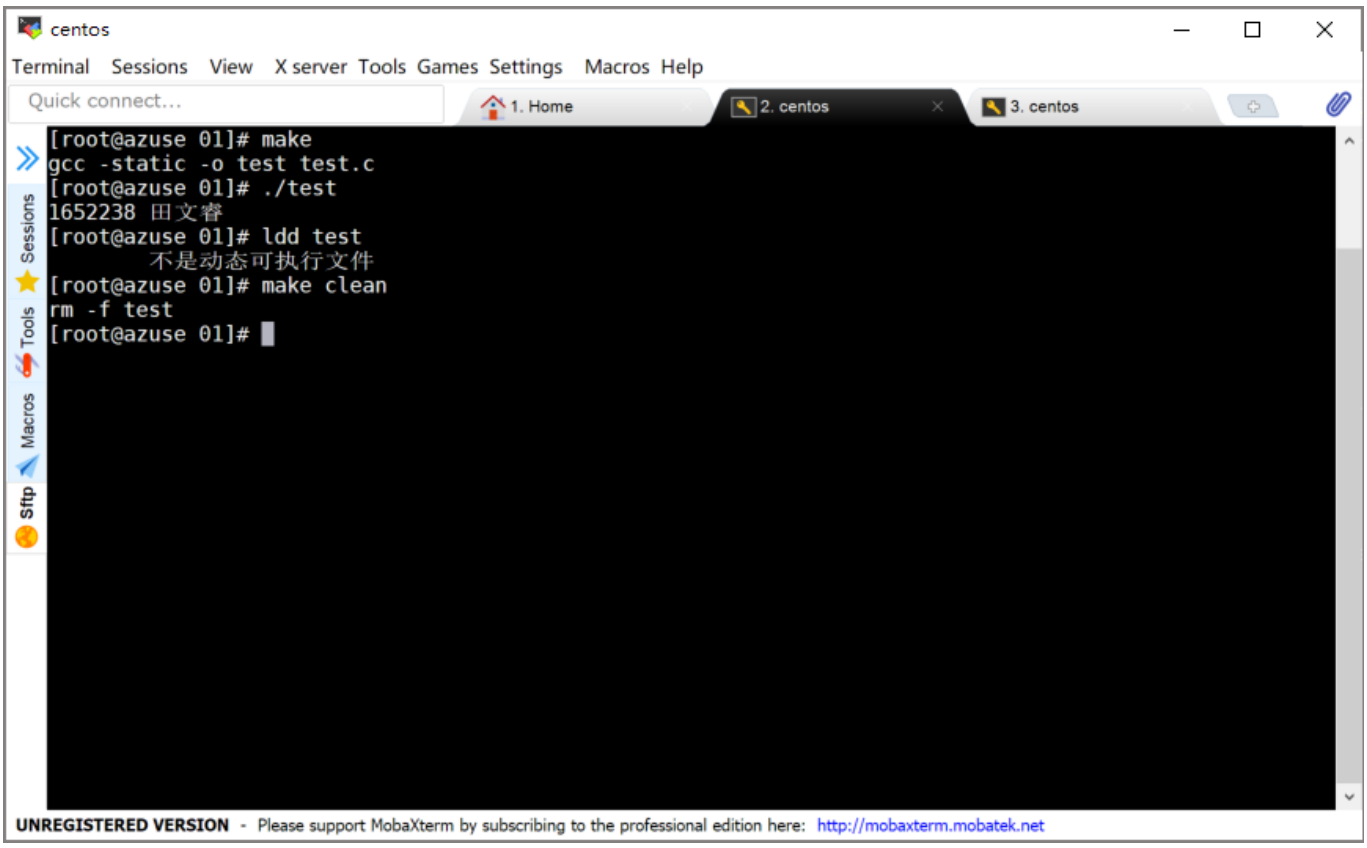
4 写出几种常用情况的静态编译测试样例

4.2 在01目录下建立test.c，使用make静态编译

makefile:

```
# makefile 01
test : test.o
	gcc -static -o test test.c
clean :
	rm -f test test.o
```

编译过程：



正常编译出静态可执行文件，使用`make clean`可以清除掉可执行文件

4.3 在子目录03下建立test.cpp，并写出满足要求的makefile文件

makefile:

```
# makefile 02
test : test.o
	g++ -o test test.cpp -static
.PHONY : clean
clean :
	rm -f test test.o
```

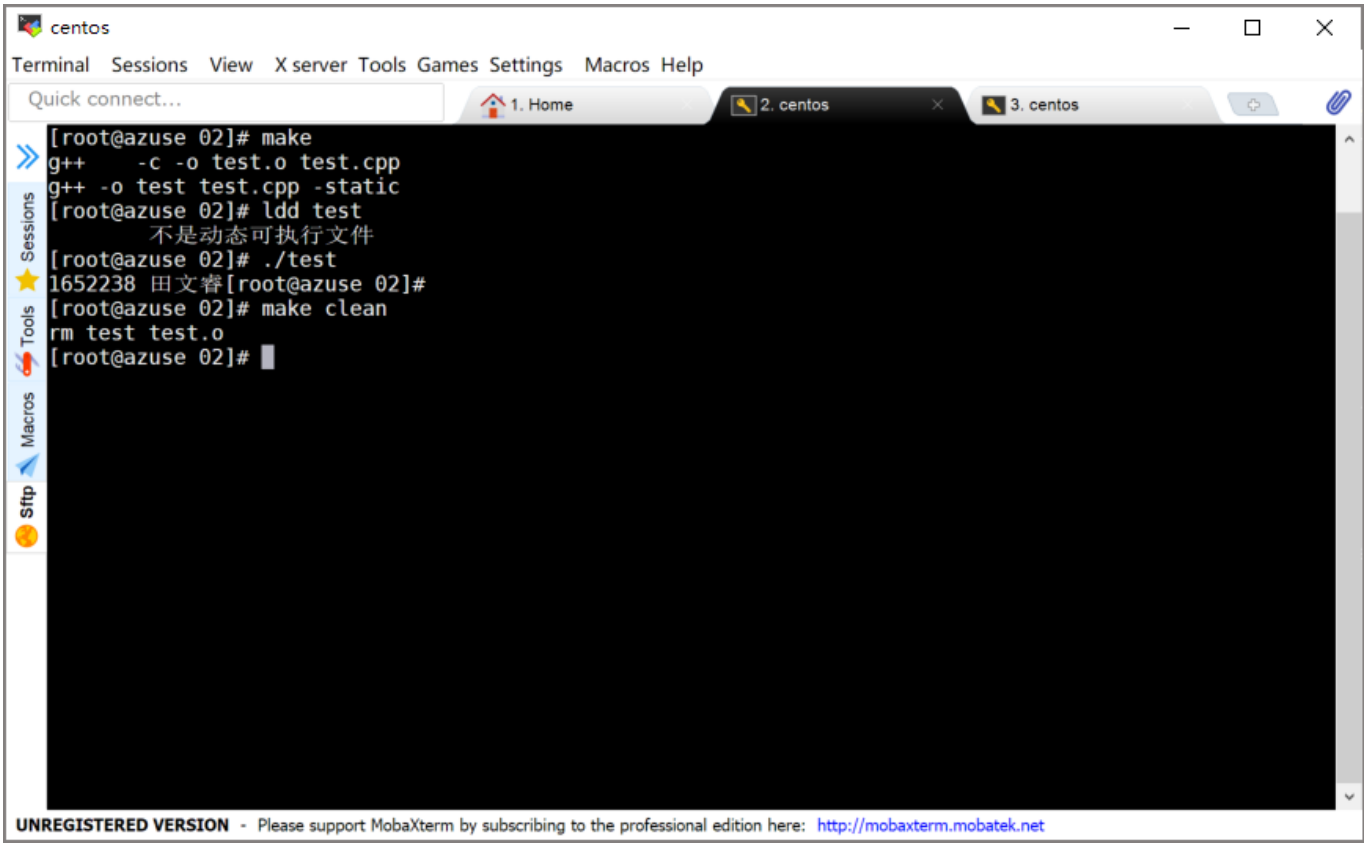
安装的静态库：

`libstdc++-static-4.8.5-28.el7.x86_64.rpm`

`glibc-static-2.17-222.el7.x86_64.rpm`

安装方法：`yum install`本地安装

编译过程：



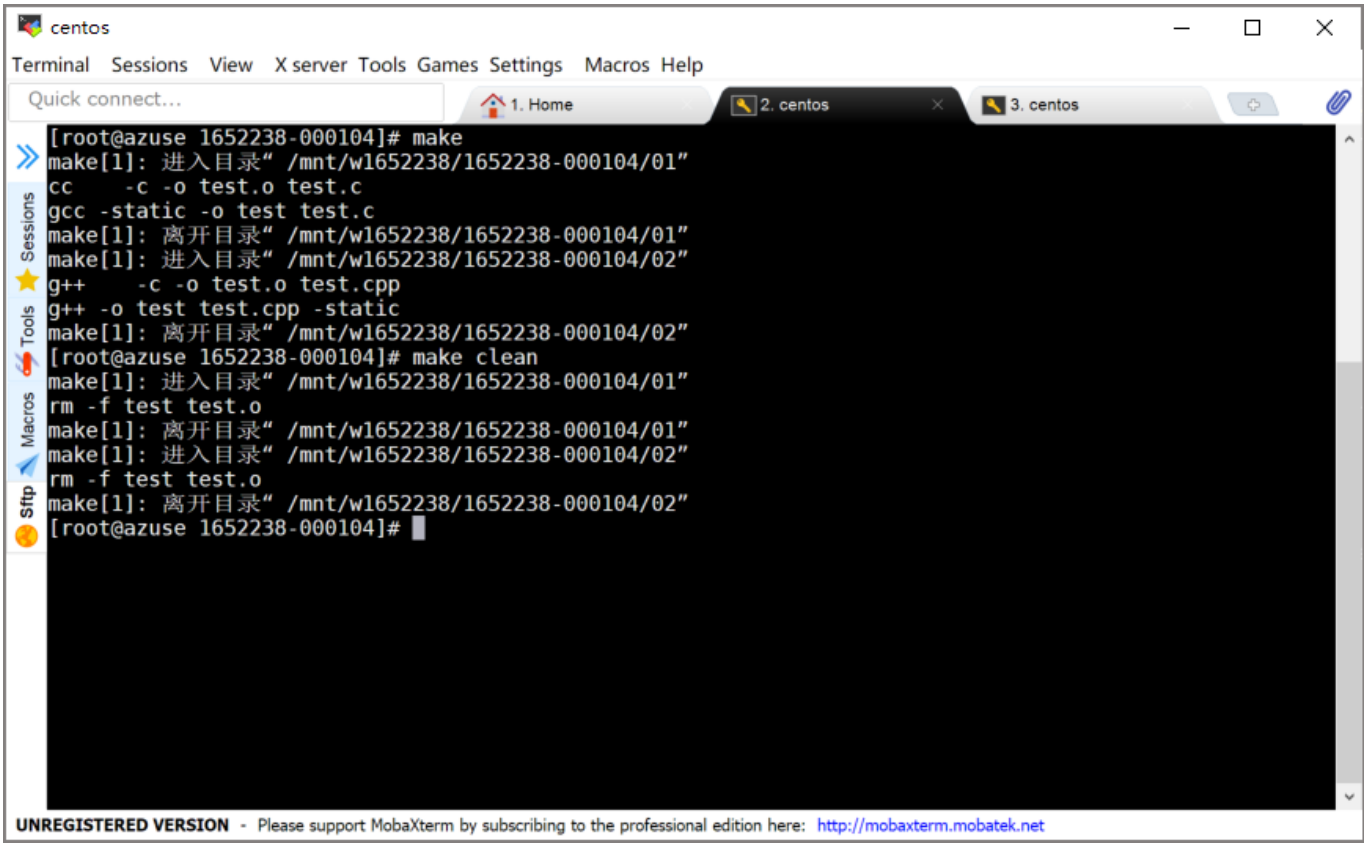
正常编译出静态可执行文件，使用`make clean`可以清楚可执行文件

4.4 1652238-000104目录下的makefile文件

可以直接将1652238-000103目录下的makefile文件拿来用

```
# makefile 1652238-000104

subdir=$(shell find . -maxdepth 1 -type d)
dirs:=$(basename $(patsubst ./%,%,$(subdir)))

.PHONY: $(dirs) clean

$(dirs):
	@for dir in $(dirs); do \
		$(MAKE) -C $$dir; \
	done

clean:
	@for dir in $(dirs); do \
		$(MAKE) -C $$dir clean; \
	done
```

编译过程：



make可以正常编译每个子目录下的文件，make clean可以清除子目录下的.o和可执行文件