扩展知识--头文件搜索

Linux中库的头文件。

首先include有两种写法，一种是#include ，另一种是#include "xxx"。这两种写法的区别是#include "xxx"会首先在当前目录下搜索头文件（不递归），如果找不到的话再去系统目录下寻找。而#include是直接去系统目录下寻找，他也是不递归找子目录里面的头文件的。

系统目录的头文件搜索规则

#include
搜索顺序是编译时指定的目录 > 环境变量指定的路径 > 系统固定搜索目录

1、编译时指定的目录

编译的时候可以通过`-I命令来指定头文件搜索目录`，比如-I /usr/local/inc就是指定在/usr/local/inc目录下寻找头文件。

2、环境变量指定的路径

对于c/c++来说是C_INCLUDE_PATH和CPLUS_INCLUDE_PATH这两个变量。可以通过`echo $C_INCLUDE_PATH`来查看当前指定的路径。可能是没有的，需要设置。

3、系统固定搜索目录

一般是`/usr/include`和`/usr/local/include`。
如果头文件在这两个目录中，那么就不需要在gcc编译的时候用-I参数指定头文件。
比如之前的MySQL数据库：

```
gcc -o db DBUtils.c -lmysqlclient -I/usr/include/mysql/ -L/usr/lib/mysql
```

这里其实可以不指定头文件所在路径，但是在代码中导入头文件时要这么写`#include <mysql/mysql.h>`，因为搜索到/usr/include的时候不会往子目录里面搜索，所以在代码的头文件要加上`mysql/`。同理，另一个固定的搜索目录也是这样。
所以，代码中的include和编译的-I参数可以搭配使用，看需求。既可以完全依靠-I参数指定头文件目录，也可以在代码中指定头文件前面的目录路径。

Linux安装Redis

1、更新镜像源

```
sudo apt-get update
```

2、下载安装redis-server

执行下面命令之后，redis安装就完成了

```
sudo apt install redis-server
```

3、查看redis-server信息

```
sudo systemctl status redis-server
```

4、配置redis（非必要）

比如配置端口号...

```
sudo vim /etc/redis/redis.conf
```

需要重启redis-server才能生效

```
sudo systemctl restart redis-server
```

5、redis-server版本查看

```
redis-server --version
或
redis-server -v
```

6、开启和关闭redis-server

```
开启：
sudo systemctl start redis-server

关闭：
sudo systemctl stop redis-server

重启：
sudo systemctl restart redis-server
```

C操作Redis

需要通过官方提供的hiredis库来实现，因此需要下载安装。

1、下载和解压库的包

```
wget https://github.com/redis/hiredis/archive/v0.14.0.tar.gz
```

```
tar -xzf v0.14.0.tar.gz
```

2、安装库

进入解压后的目录，执行一下命令

```
make && make install
```

执行完之后就会自动把libhiredis.so放到 `/usr/local/lib/` 中，把hiredis.h放到 `/usr/local/inlcude/hiredis/` 中

3、使用库

上面执行完成之后就可以编写相关的C程序了。
如下代码：

```
#include <iostream>
//导入hiredis的头文件，这一个就可以了
#include <hiredis/hiredis.h>

using namespace std;

int main()
{
    redisContext *c = redisConnect("127.0.0.1", 6379);
    if (c->err)
    {
        redisFree(c);
        cout << "connect to redis fail" << endl;
        return 1;
    }
    cout << "connect to redis success" << endl;
    redisReply *r = (redisReply *)redisCommand(c, "get name");
    cout << r->str << endl;
    return 0;
}
```

编译代码

```
g++ rediscon.cpp -o rediscon -L/usr/local/lib/ -lhiredis
```

头文件-->因为它在头文件固定搜索的`/usr/local/include`目录中，所以只要在代码中的导入头文件加上一层目录即可。这样无需再编译指令中加`-I`参数
库名-->hiredis。如果不知道，可以在`/usr/local/lib`目录中查看。

```
kiko@Hkiko:/usr/local/lib$ ls -al
总用量 1112
drwxr-xr-x  5 root root    4096 6月   7 09:06 .
drwxr-xr-x 10 root root    4096 6月   5 10:01 ..
-rw-rw-r--  1 kiko kiko  762870 6月   5 10:42 libhiredis.a
lrwxrwxrwx  1 root root      18 6月   7 09:06 libhiredis.so -> libhiredis.so.0.14
-rwxrwxr-x  1 kiko kiko  350672 6月   5 10:42 libhiredis.so.0.14
drwxr-xr-x  2 root root    4096 6月   5 10:56 pkgconfig
```


库路径-->因为hiredis的库在`/usr/local/lib`中，不在动态库默认搜索路径中，所以要在g++编译时指定库的路径。当然也可以将其设置到动态库的默认搜索目录中。
但是，经过`ldd`命令查看之后发现，不需要指定-L也是可以的，因为它默认也把库安装在了`/lib`的一个子目录中，/lib这个目录也是一个动态库的默认搜索路径，所以无论加不加`-L`参数指定，其实都是在/lib目录中找到的动态库。

```
kiko@Hkiko:~/lesson/myredis$ g++ rediscon.cpp -o rediscon -L/usr/local/lib/ -lhiredis  -I/usr/local/include/hiredis
kiko@Hkiko:~/lesson/myredis$ ldd rediscon
        linux-vdso.so.1 (0x00007ffc405b9000)
        libhiredis.so.0.14 => /lib/x86_64-linux-gnu/libhiredis.so.0.14 (0x00007f1c44888000)
        libstdc++.so.6 => /lib/x86_64-linux-gnu/libstdc++.so.6 (0x00007f1c446a6000)
        libc.so.6 => /lib/x86_64-linux-gnu/libc.so.6 (0x00007f1c444b4000)
        libm.so.6 => /lib/x86_64-linux-gnu/libm.so.6 (0x00007f1c44365000)
        /lib64/ld-linux-x86-64.so.2 (0x00007f1c448b4000)
        libgcc_s.so.1 => /lib/x86_64-linux-gnu/libgcc_s.so.1 (0x00007f1c4434a000)
```


所以最终命令可以写成：

```
g++ rediscon.cpp -o rediscon -lhiredis
```