CLion调试redis6源码（全网首发）

背景

clion使用cmake来管理编译redis源码，而redis源码本身使用原生的make，因此直接将redis源码导入clion无法直接运行，需要配置cmake。

写c程序大体步骤为：

1).用编辑器编写源代码，如.c文件。

2).用编译器编译代码生成目标文件，如.o。

3).用链接器连接目标代码生成可执行文件，如.exe。

但如果源文件太多，一个一个编译时就会特别麻烦，于是人们想到，为什么不设计一种类似批处理的程序，来批处理编译源文件呢，于是就有了make工具，它是一个自动化编译工具，你可以使用一条命令实现完全编译。但是你需要编写一个规则文件，make依据它来批处理编译，这个文件就是makefile，所以编写makefile文件也是一个程序员所必备的技能。

对于一个大工程，编写makefile实在是件复杂的事，于是人们又想，为什么不设计一个工具，读入所有源文件之后，自动生成makefile呢，于是就出现了cmake工具，它能够输出各种各样的makefile或者project文件,从而帮助程序员减轻负担。但是随之而来也就是编写cmakelist文件，它是cmake所依据的规则。所以在编程的世界里没有捷径可走，还是要脚踏实地的。

所以流程如下：

一个程序，在linux下运行，你要写一份makefile，如果要移植到到其他平台，这个makefile就用不了了，需要再写一份。

所以，为了跨平台，出现了cmake，cmake是让程序员用统一的语法来写cmake文件，然后cmake会帮助我们生成对应的平台下的makefile。

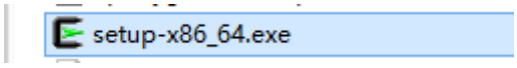
所以我们选用cygwin来在window下模拟linux环境，cygwin里默认带有cmake编译工具。

一、安装配置cygwin

1）打开官网:https://cygwin.com/install.html

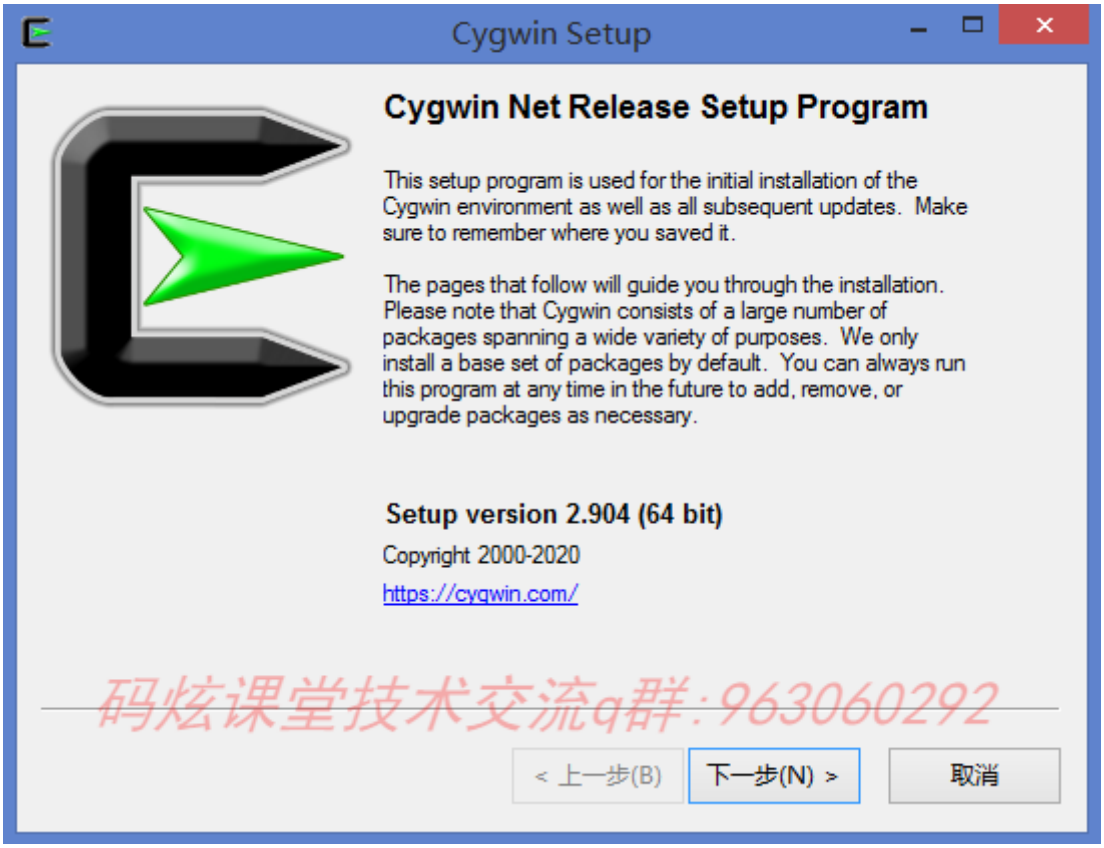
2）进入上图的install链接（下图），根据自己的电脑选择32位还是64位

下载https://cygwin.com/setup-x86_64.exe（这是64位的，如果下载32位的，后面的配置和64位是一样的）

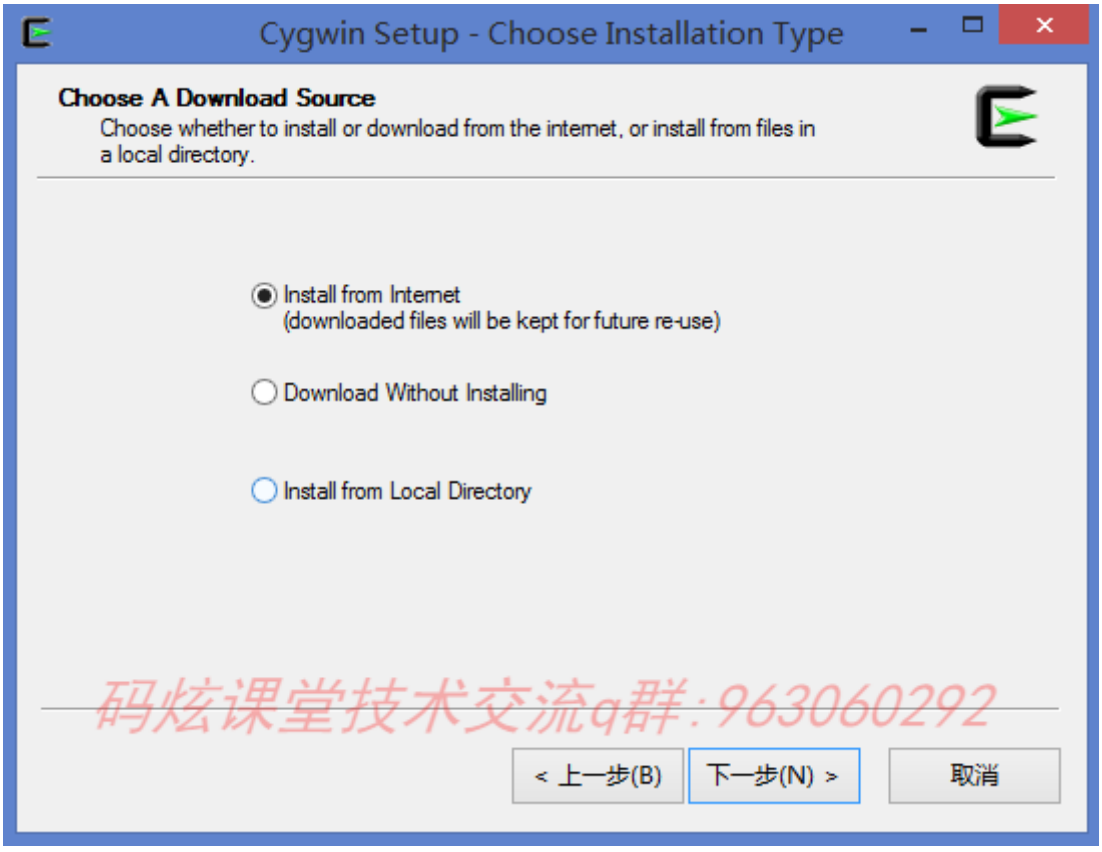


3）安装

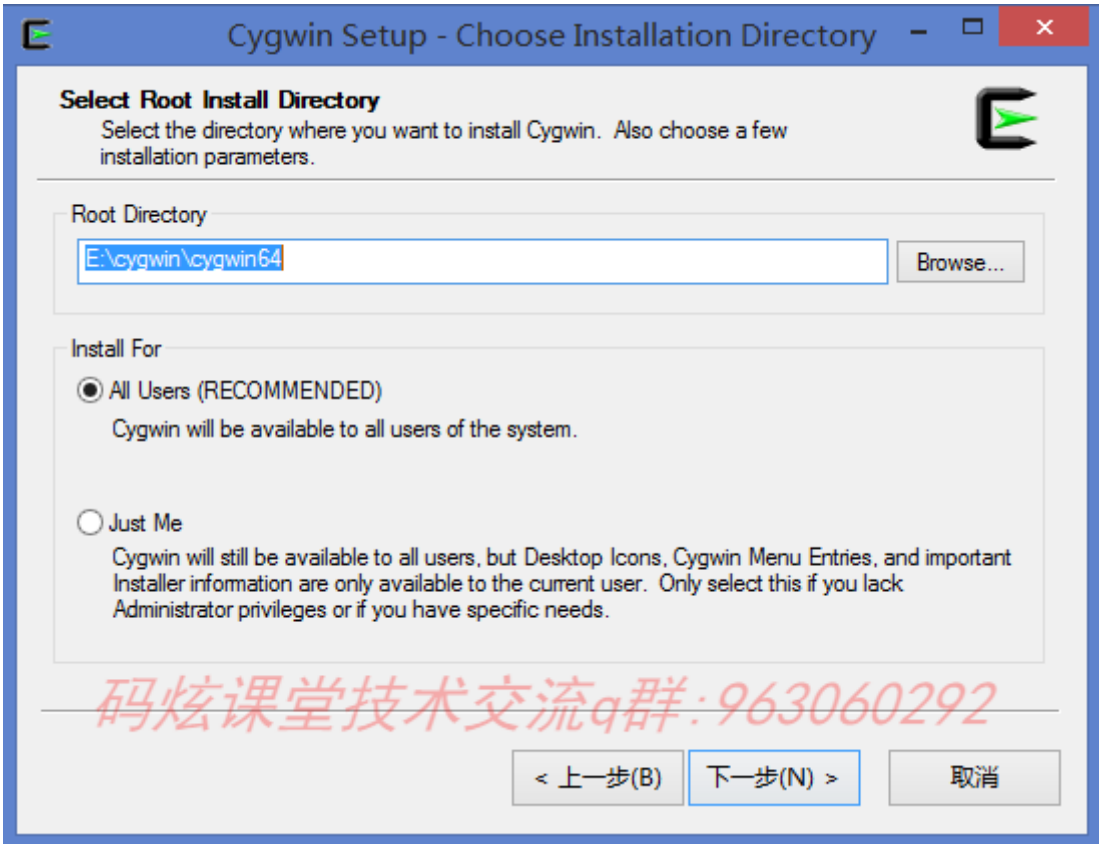
双击安装即可



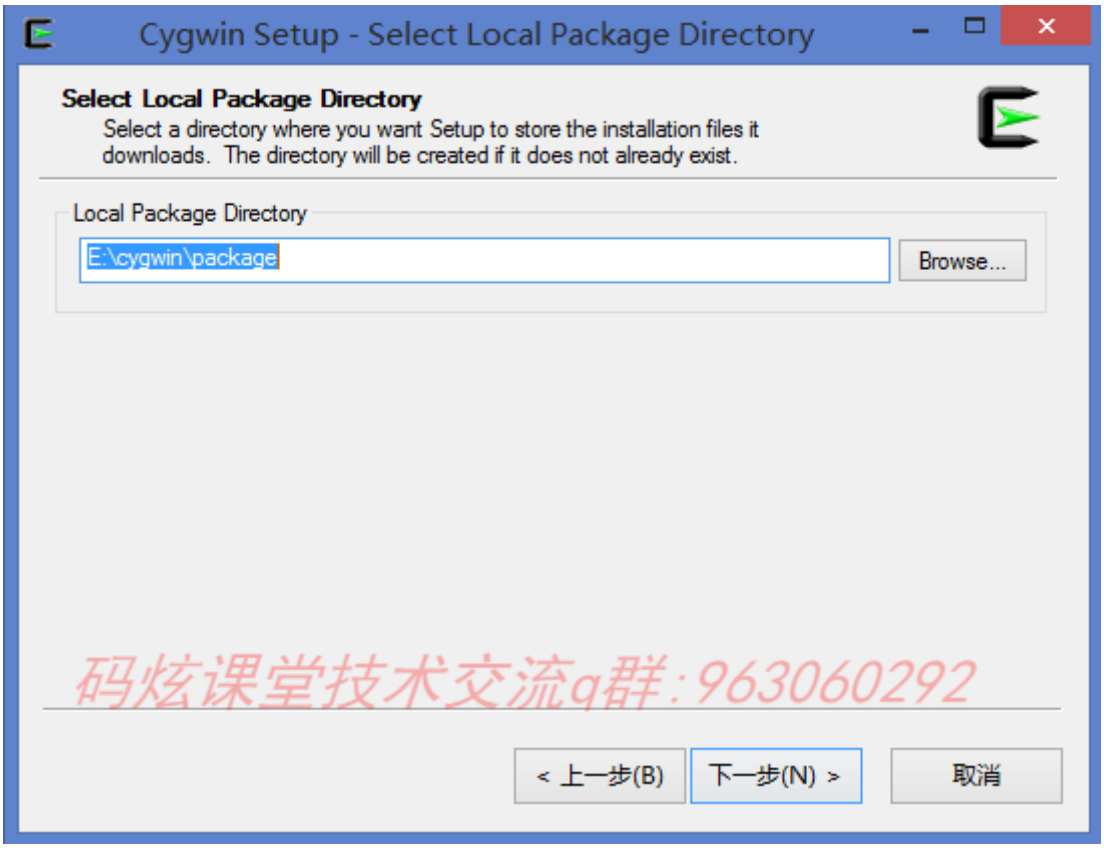
下一步->install from Internet -> 设置下载安装路径 -> direct connection -> 选择下载源（可以添加网易镜像站：http://mirrors.163.com/cygwin/ [参考http://mirrors.163.com/.help/cygwin.html] ）
接来下选择安装的模块，分别搜索wget、 gcc-core、gcc-g++、make、gdb、binutils，以上所有项目都在 devel 文件夹下。



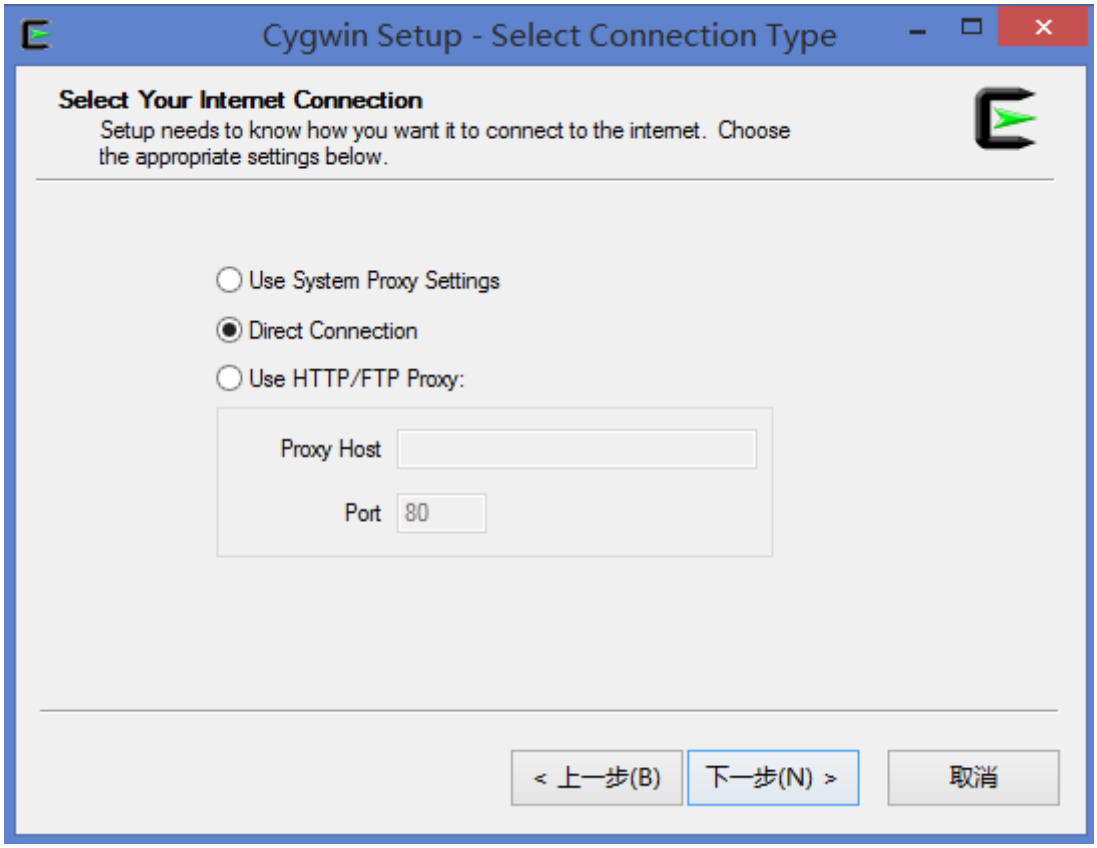
【下一步】


Cygwin Setup - Choose Installation Directory
Select Root Install Directory
Select the directory where you want to install Cygwin. Also choose a few installation parameters.
Root Directory
E:\cygwin\cygwin64
Browse...
Install For
All Users (RECOMMENDED)
Cygwin will be available to all users of the system.
Just Me
Cygwin will still be available to all users, but Desktop Icons, Cygwin Menu Entries, and important Installer information are only available to the current user. Only select this if you lack Administrator privileges or if you have specific needs.
码炫课堂技术交流q群:963060292
< 上一步(B)
下一步(N) >
取消

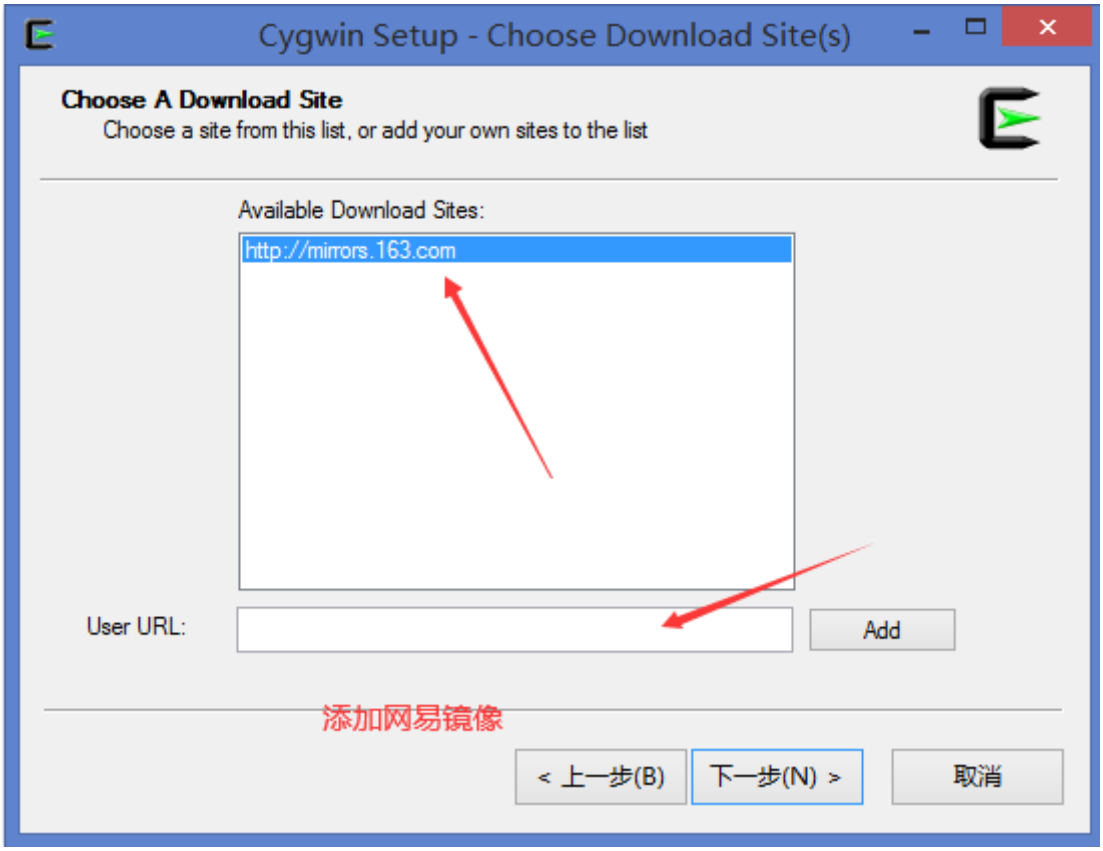
继续【下一步】


Cygwin Setup - Select Local Package Directory
Select Local Package Directory
Select a directory where you want Setup to store the installation files it downloads. The directory will be created if it does not already exist.
Local Package Directory
E:\cygwin\package
Browse...
码炫课堂技术交流q群:963060292
< 上一步(B)
下一步(N) >
取消

继续【下一步】

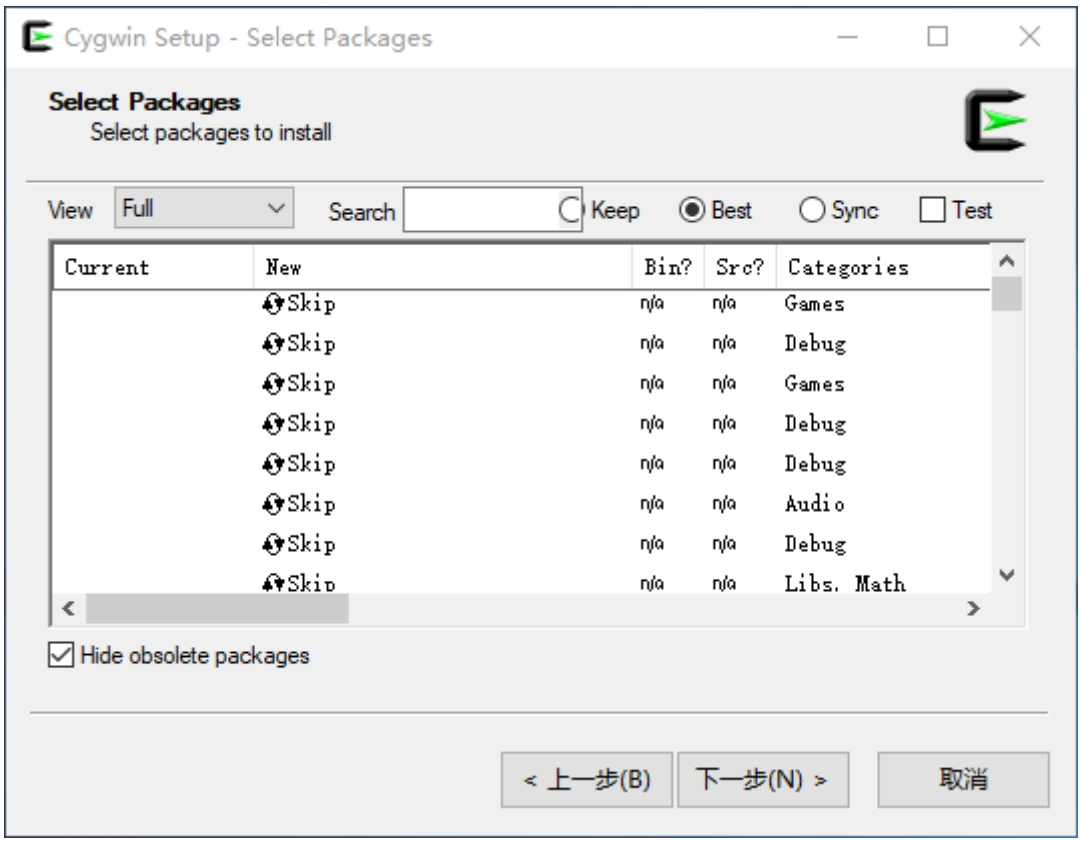


继续【下一步】

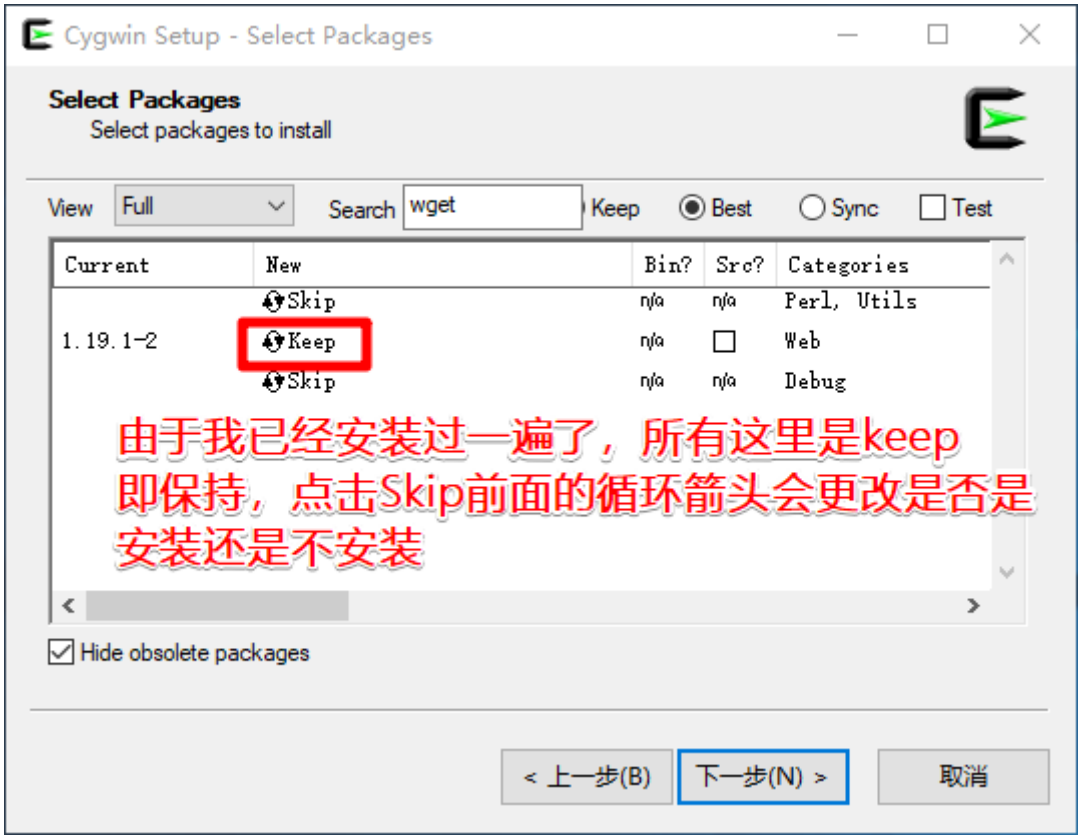


添加网易镜像后继续【下一步】

下图中选择要安装的包（我们需要安装wget、 gcc-core、gcc-g++、make、gdb、binutils ）：



在search框中搜索以上工具集，如下图：

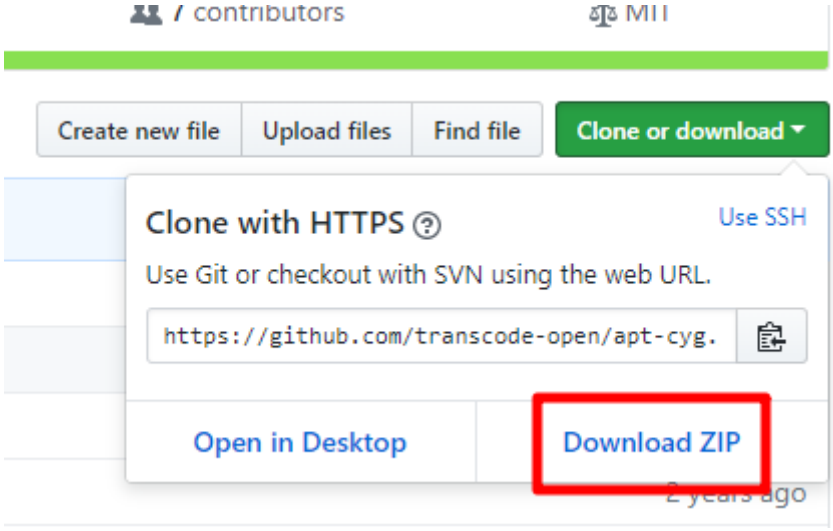


还要安装一个apt-cyg（相当于是linux下的yum，cygwin下没有yum命令）

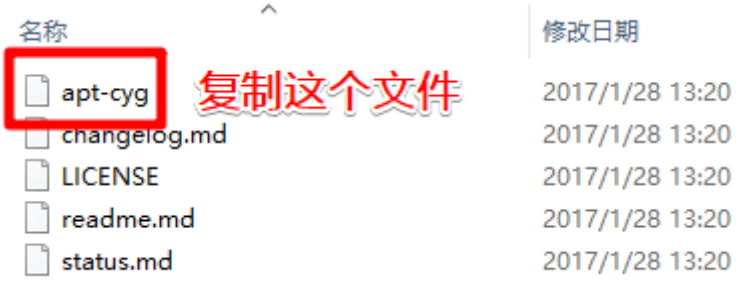
为什么要安装apt-cyg?

安装了apt-cyg就能向使用Ubuntu一样使用apt-get install/remove命令安装卸载软件了，非常方便。

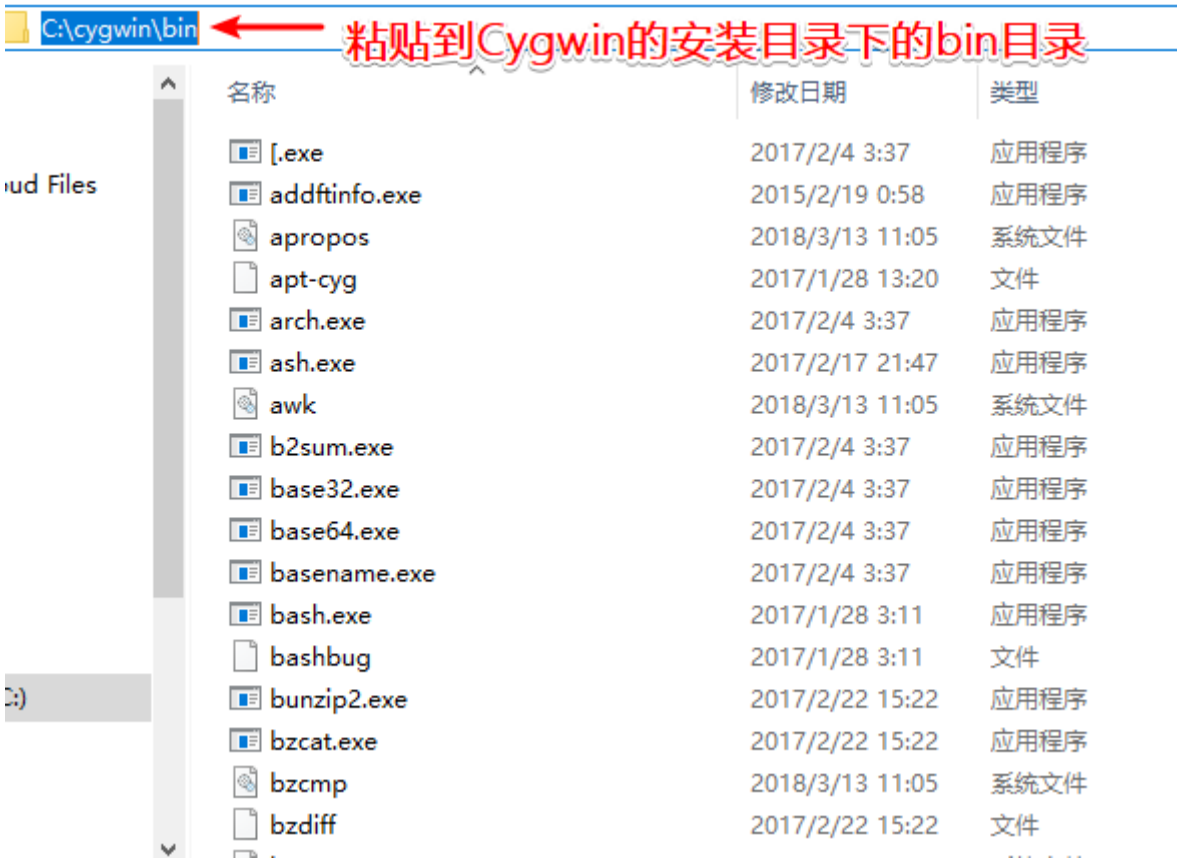
GitHub下载脚本：https://github.com/transcode-open/apt-cyg



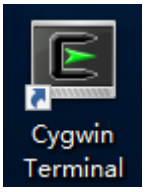
解压刚刚下载的zip文件：



复制apt-cyg，粘贴到cygwin的安装目录的bin目录下：

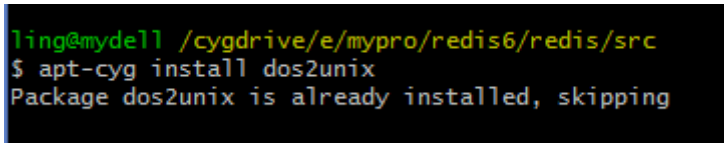


打开Cygwin



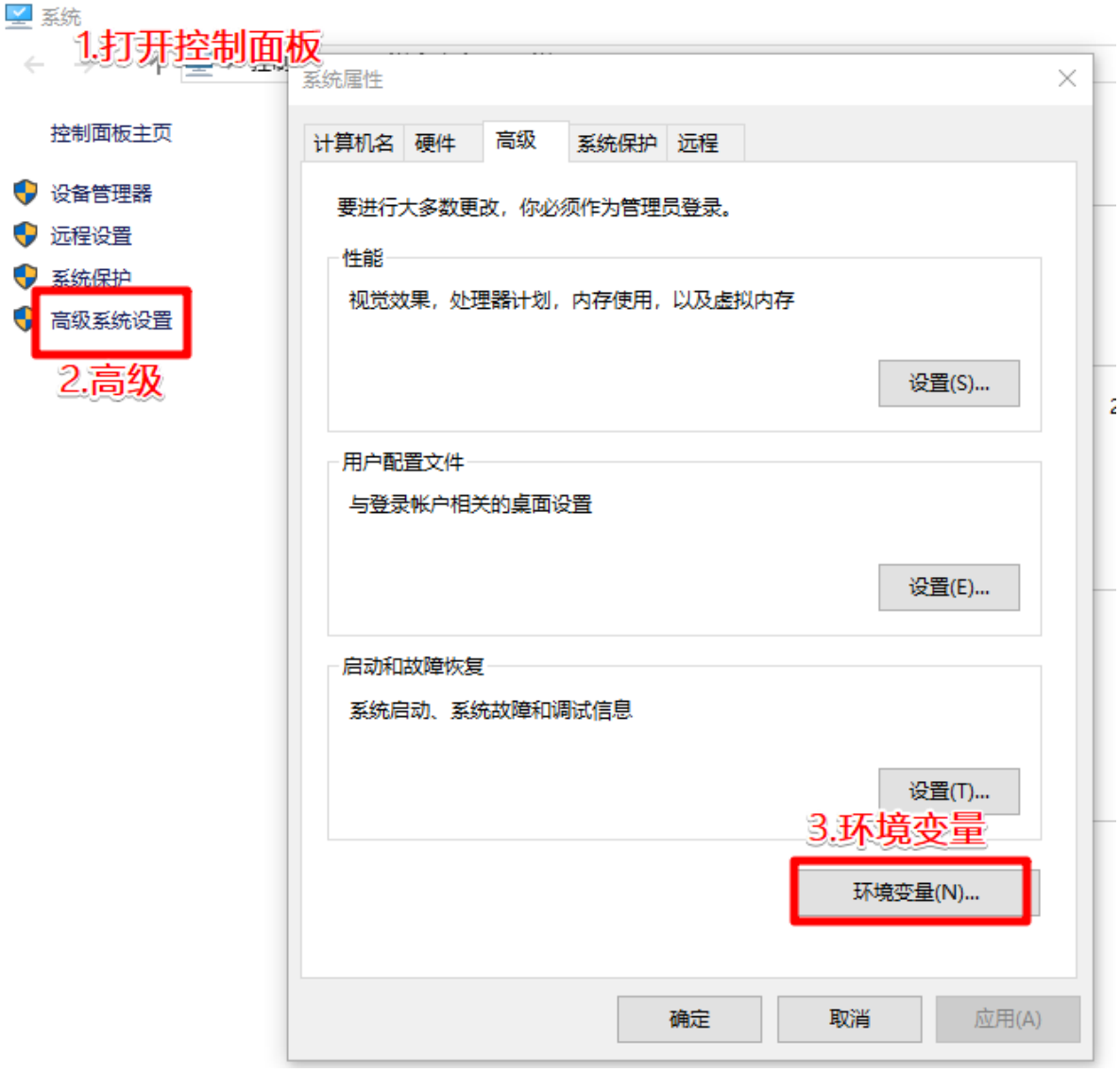
输入命令：

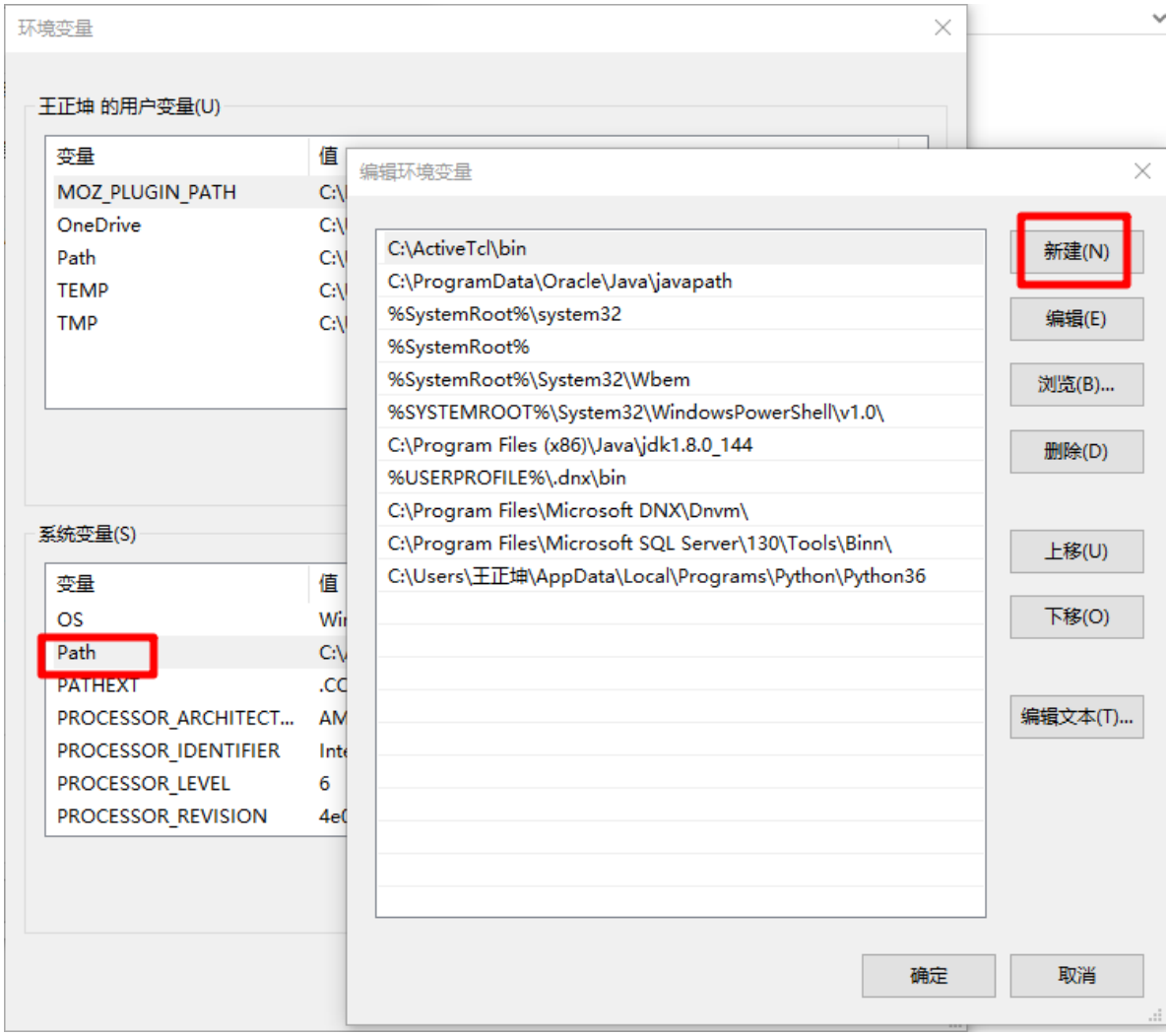
```
##该命令后面会用到
apt-cyg install dos2unix
```

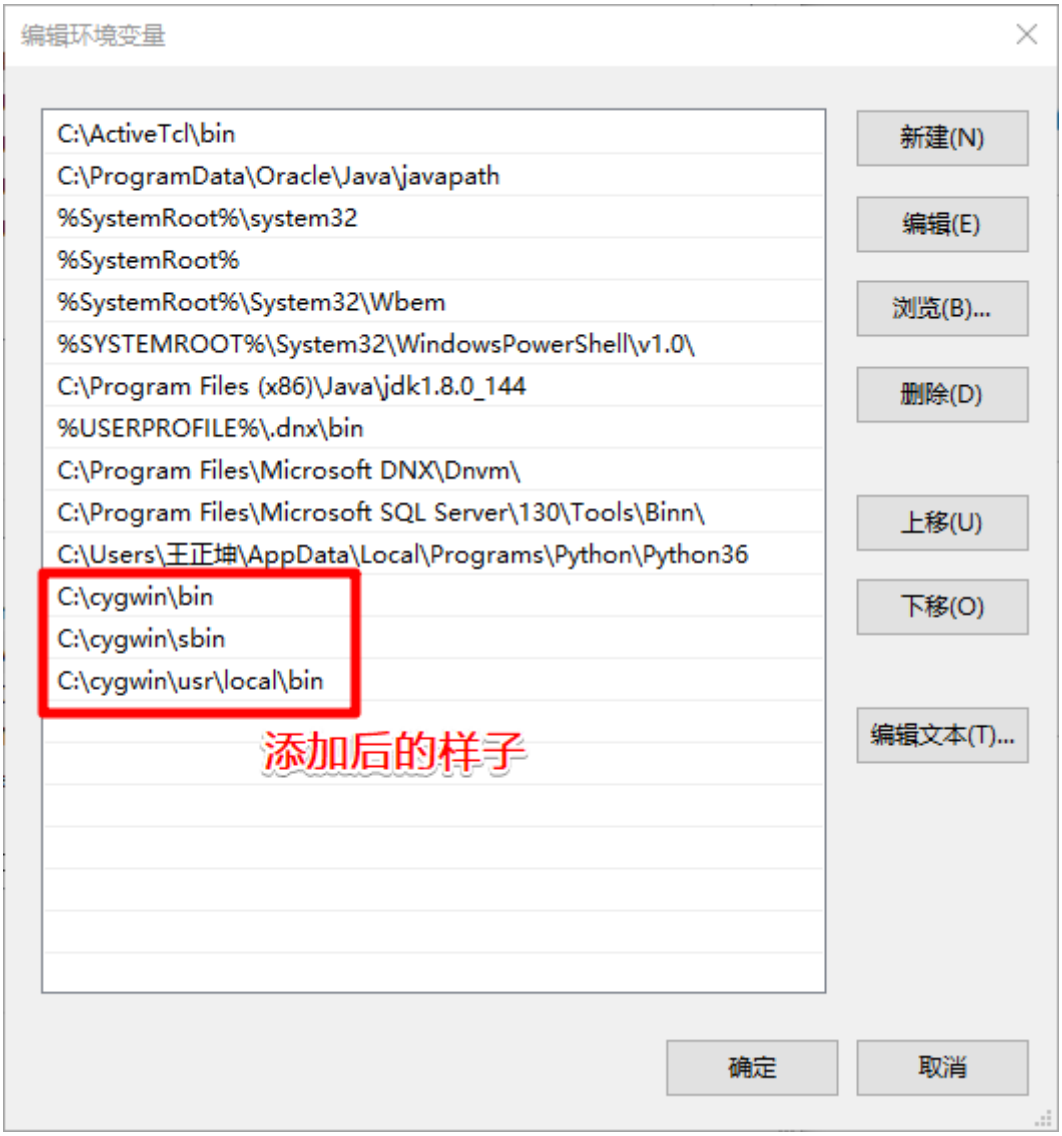


由于我已经安装过，所以这里就跳过了，这句命令测试apt-cyg是否成功运行。

4）添加环境变量以便在cmd或者powershell中可以使用Linux命令




环境变量
王正坤 的用户变量(U)
变量
值
MOZ_PLUGIN_PATH
OneDrive
Path
TEMP
TMP
系统变量(S)
变量
值
OS
Path
PATHEXT
PROCESSOR_ARCHITECT...
PROCESSOR_IDENTIFIER
PROCESSOR_LEVEL
PROCESSOR_REVISION
编辑环境变量
C:\ActiveTcl\bin
C:\ProgramData\Oracle\Java\javapath
%SystemRoot%\system32
%SystemRoot%
%SystemRoot%\System32\Wbem
%SYSTEMROOT%\System32\WindowsPowerShell\v1.0\
C:\Program Files (x86)\Java\jdk1.8.0_144
%USERPROFILE%\.dnx\bin
C:\Program Files\Microsoft DNX\Dnvm\
C:\Program Files\Microsoft SQL Server\130\Tools\Binn\
C:\Users\王正坤\AppData\Local\Programs\Python\Python36
新建(N)
编辑(E)
浏览(B)...
删除(D)
上移(U)
下移(O)
编辑文本(T)...
确定
取消



以上环境变量内容请根据自己的实际安装地址进行配置。

二、安装clion

CLion 是 JetBrains 推出的全新的 C/C++ 跨平台集成开发环境。习惯了用idea的java程序员用clion来调试redis源码应该比较舒适，比起vc要舒服多了。

什么？？我搞java的现在让我调试c代码？？

额。。。谁规定java程序员不能精通C/C++的？况且redis本来就是c写的，你要研究redis源码，能不把c重新拾起来吗？

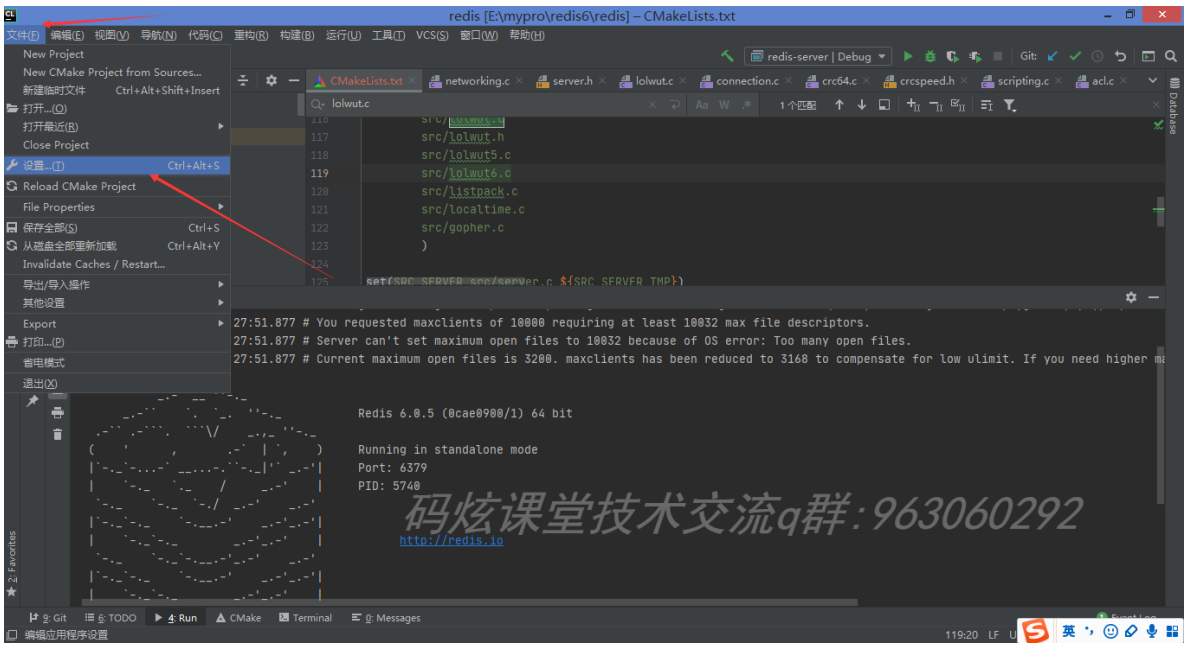
废话少说，走起！

1）、下载并安装中文破解版clion

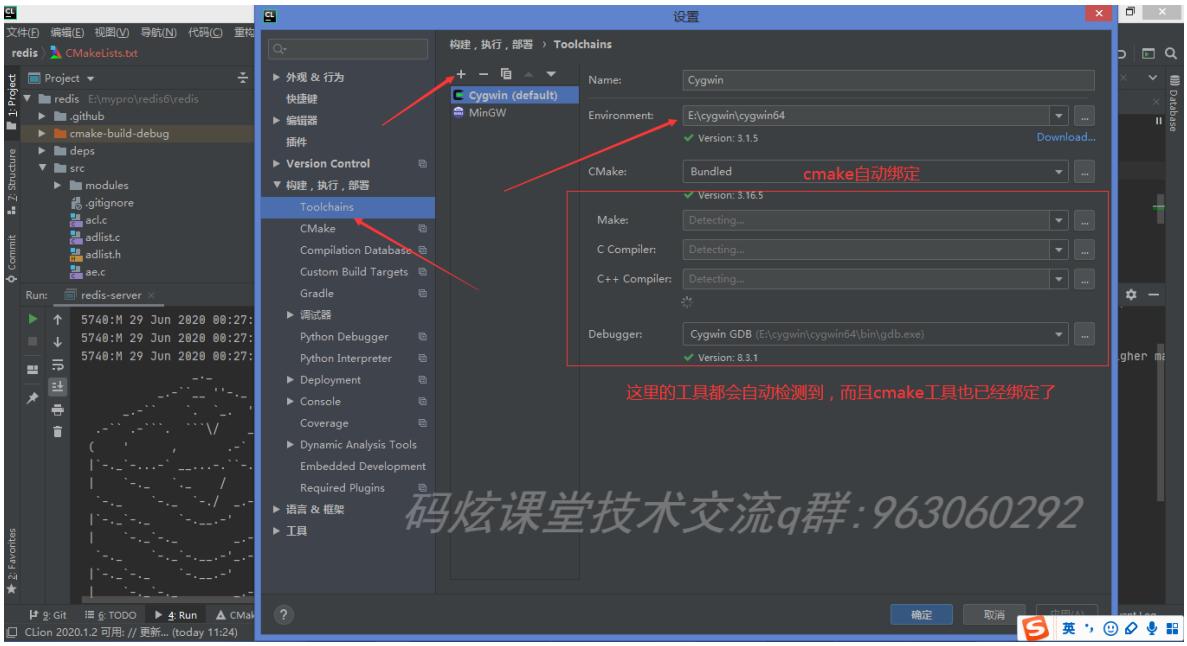
链接：https://pan.baidu.com/s/1o0HEOdqhWYBsfqxKgOZWog
提取码：cp5t

2）、安装好之后，就需要在clion中配置cygwin编译器

依次点击：【文件】->【设置】



3）、打开配置对话框，选择【构建，执行，部署】->【Toolchains】->点击“+”号，选择【Cygwin】，配置刚刚安装好的Cygwin环境，下面cmake等工具都会自动检测到。



注：除了Cygwin以外，这里还可以配置MinGW,VS等。

如果使用MinGW的话，安装好MinGW后会缺少很多文件，所以本教程使用Cygwin。

配置完成之后开始下载redis源码了。

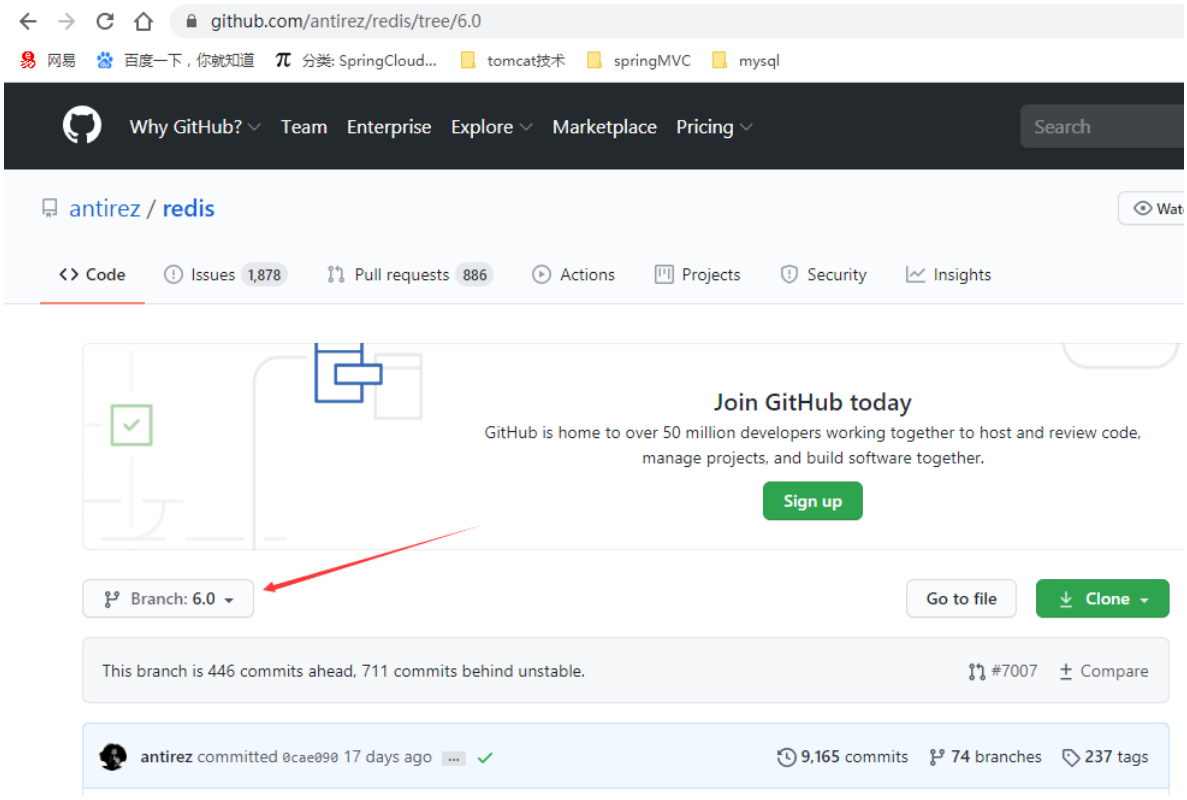
三、clion中导入redis源码

1）、我们从git上拉取当前最新的稳定版6.x版本。

建议从官方仓库 https://github.com/antirez/redis Fork 出属于自己的仓库。

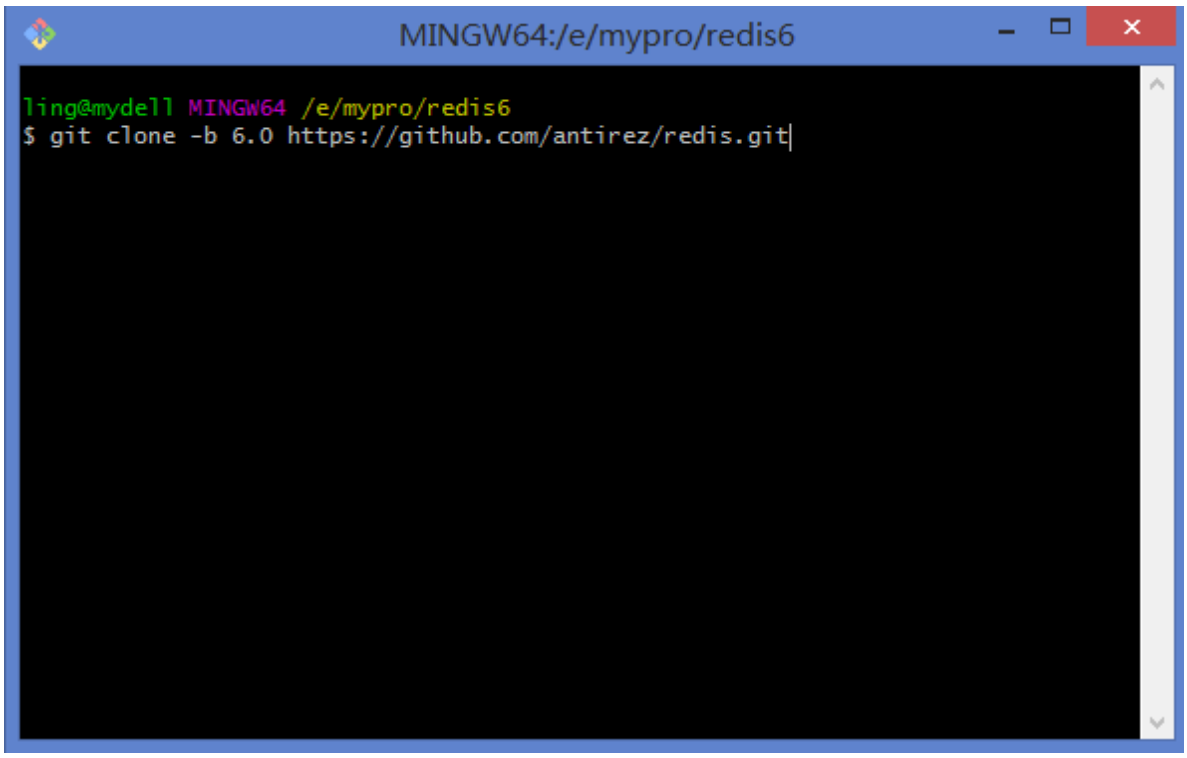
- 为什么要 Fork ？既然开始阅读、调试源码，我们可能会写一些注释，有了自己的仓库，可以进行自由的提交。

- 本文使用的 Redis 版本为最新的 `6.0.5` 。



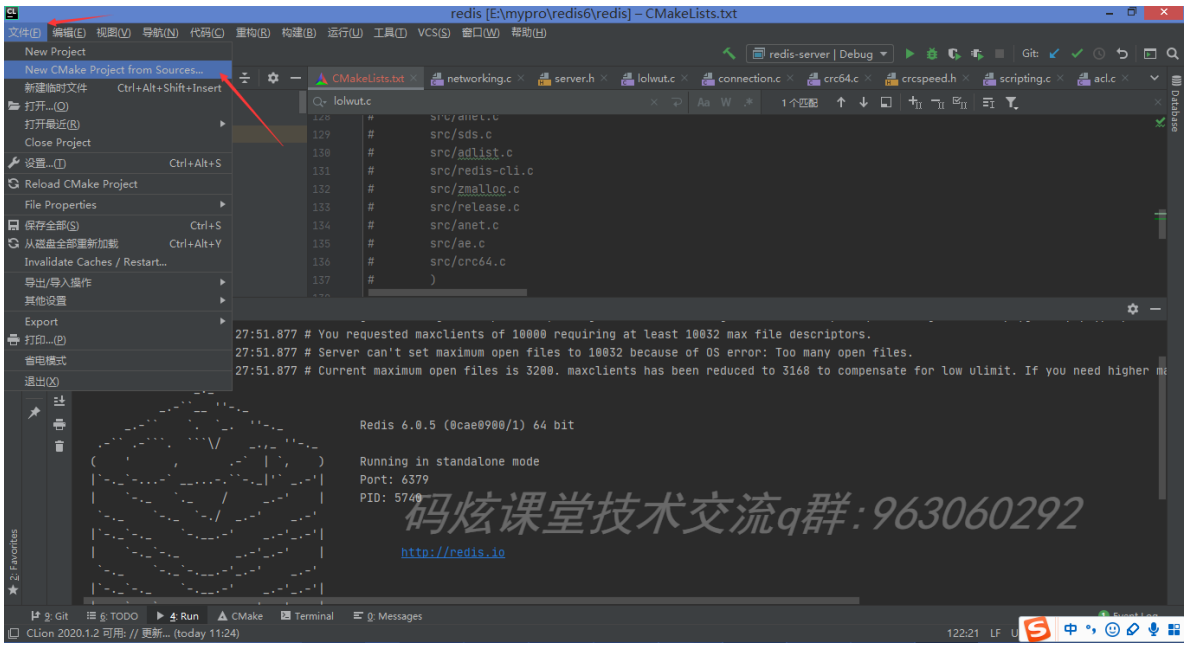
2）、打开gitbush，执行：

```
git clone -b 6.0 https://github.com/antirez/redis.git
```

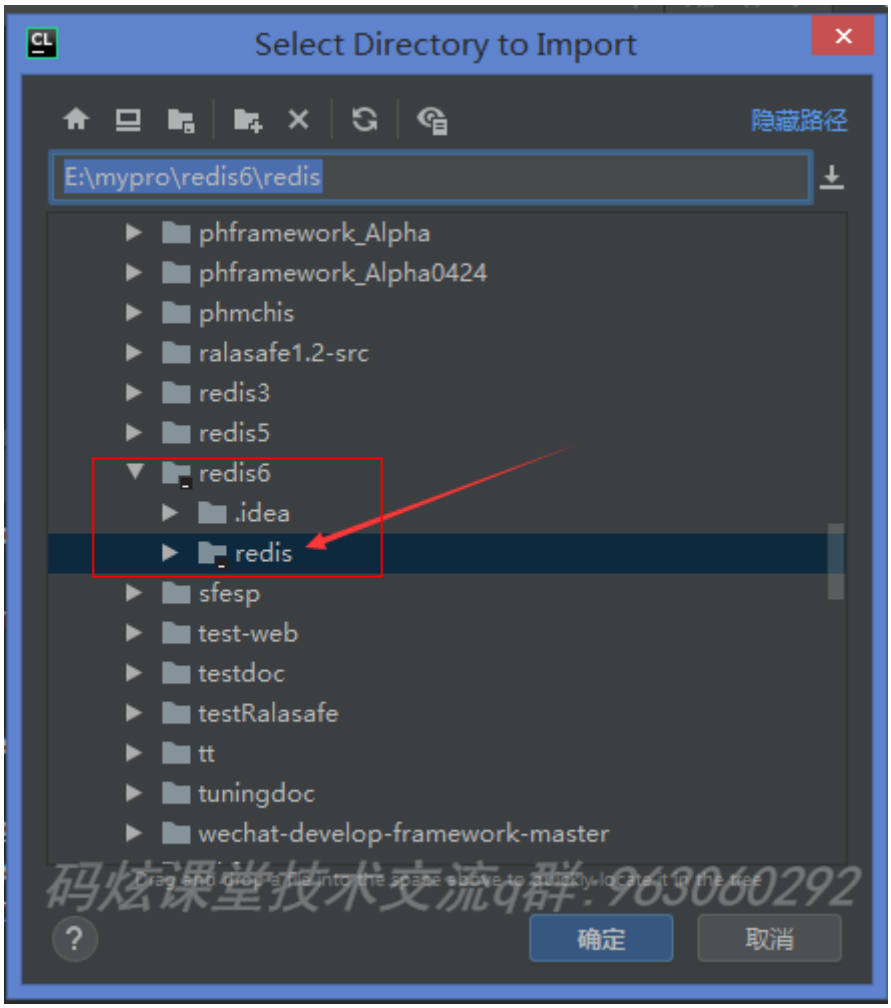


3）、redis6源码导入clion中

【文件】->【New CMake Project from Sources...】



选择刚clone的redis源码，点击【确定】，如下图：



四、修改CMakeLists.txt文件

我们需要修改5个CMakeLists.txt文件

```
./CMakeLists.txt
./deps/CMakeLists.txt
./deps/linenoise/CMakeLists.txt
./deps/lua/CMakeLists.txt
./src/modules/CMakeLists.txt
```

./CMakeLists.txt修改如下：

```
cmake_minimum_required(VERSION 3.0 FATAL_ERROR)
project(redis VERSION 6.0)

if (NOT CMAKE_BUILD_TYPE)
    message(STATUS "No build type defined; defaulting to 'Debug'")
    set(CMAKE_BUILD_TYPE "Debug" CACHE STRING
            "The type of build. Possible values are: Debug, Release,
RelWithDebInfo and MinSizeRel.")
endif()

message(STATUS "Host is: ${CMAKE_HOST_SYSTEM}.  Build target is:
${CMAKE_SYSTEM}")
get_filename_component(REDIS_ROOT "${CMAKE_CURRENT_SOURCE_DIR}" ABSOLUTE)
message(STATUS "Project root directory is: ${REDIS_ROOT}")

# Just for debugging when handling a new platform.
if (false)
    message("C++ compiler supports these language features:")
    foreach(i ${CMAKE_CXX_COMPILE_FEATURES})
        message("  ${i}")
    endforeach()
endif()

message(STATUS "Generating release.h...")
execute_process(
        COMMAND sh -c ./mkreleasehdr.sh
        WORKING_DIRECTORY ${REDIS_ROOT}/src/
)

add_subdirectory(deps)
add_subdirectory(src/modules)

set(SRC_SERVER_TMP
        src/crcspeed.c
        src/crcspeed.h
        src/sha256.c
        src/sha256.h
        src/connection.c
        src/connection.h
        src/acl.c
        src/timeout.c
        src/tracking.c
        src/tls.c
        src/adlist.c
        src/ae.c
        src/anet.c
        src/dict.c
```

```
        src/sds.c
        src/zmalloc.c
        src/lzf_c.c
        src/lzf_d.c
        src/pqsort.c
        src/zipmap.c
        src/sha1.c
        src/ziplist.c
        src/release.c
        src/networking.c
        src/util.c
        src/object.c
        src/db.c
        src/replication.c
        src/rdb.c
        src/t_string.c
        src/t_list.c
        src/t_set.c
        src/t_zset.c
        src/evict.c
        src/defrag.c
        src/module.c
        src/quicklist.c
        src/expire.c
        src/childinfo.c
        src/redis-check-aof.c
        src/redis-check-rdb.c
        src/lazyfree.c
        src/geohash.c
        src/rax.c
        src/geohash_helper.c
        src/siphash.c
        src/geo.c
        src/t_hash.c
        src/config.c
        src/aof.c
        src/pubsub.c
        src/multi.c
        src/debug.c
        src/sort.c
        src/intset.c
        src/syncio.c
        src/cluster.c
        src/crc16.c
        src/endianconv.c
        src/slowlog.c
        src/scripting.c
        src/bio.c
        src/rio.c
        src/rand.c
        src/memtest.c
        src/crc64.c
        src/bitops.c
        src/sentinel.c
        src/notify.c
        src/setproctitle.c
        src/blocked.c
        src/hyperloglog.c
```

```
        src/latency.c
        src/sparkline.c
        src/t_stream.c
        src/lolwut.c
        src/lolwut.h
        src/lolwut5.c
        src/lolwut6.c
        src/listpack.c
        src/localtime.c
        src/gopher.c
        )

set(SRC_SERVER src/server.c ${SRC_SERVER_TMP})

set(SRC_CLI
        src/anet.c src/sds.c src/adlist.c src/redis-cli.c src/zmalloc.c
src/release.c src/anet.c src/ae.c src/crc64.c
        )
if (${CMAKE_SYSTEM_NAME} MATCHES "Linux")
    # better not to work with jemalloc
endif()

add_executable(redis-server ${SRC_SERVER})
add_executable(redis-cli ${SRC_CLI})

set_property(TARGET redis-server PROPERTY C_STANDARD 99)
set_property(TARGET redis-server PROPERTY CXX_STANDARD 11)
set_property(TARGET redis-server PROPERTY CXX_STANDARD_REQUIRED ON)

set_property(TARGET redis-cli PROPERTY C_STANDARD 99)
set_property(TARGET redis-cli PROPERTY CXX_STANDARD 11)
set_property(TARGET redis-cli PROPERTY CXX_STANDARD_REQUIRED ON)

target_include_directories(redis-server
        PRIVATE ${REDIS_ROOT}/deps/hiredis
        PRIVATE ${REDIS_ROOT}/deps/linenoise
        PRIVATE ${REDIS_ROOT}/deps/lua/src
        )

target_include_directories(redis-cli
        PRIVATE ${REDIS_ROOT}/deps/hiredis
        PRIVATE ${REDIS_ROOT}/deps/linenoise
        PRIVATE ${REDIS_ROOT}/deps/lua/src
        )

target_link_libraries(redis-server
        PRIVATE pthread
        PRIVATE m
        PRIVATE lua
        PRIVATE linenoise
        PRIVATE hiredis
        )

target_link_libraries(redis-cli
        PRIVATE pthread
        PRIVATE m
        PRIVATE linenoise
        PRIVATE hiredis
```

```
        )

link_directories(deps/hiredis/ deps/linenoise/ diredeps/lua/src)
```

./deps/CMakeLists.txt修改如下：

```
add_subdirectory(hiredis)
add_subdirectory(linenoise)
add_subdirectory(lua)
```

./deps/linenoise/CMakeLists.txt修改如下：

```
add_library(linenoise linenoise.c)
```

./deps/lua/CMakeLists.txt修改如下：

```
set(LUA_SRC
        src/lauxlib.c
        src/liolib.c
        src/lopcodes.c
        src/lstate.c
        src/lobject.c
        src/print.c
        src/lmathlib.c
        src/loadlib.c
        src/lvm.c
        src/lfunc.c
        src/lstrlib.c
        src/lua.c
        src/linit.c
        src/lstring.c
        src/lundump.c
        src/luac.c
        src/ltable.c
        src/ldump.c
        src/loslib.c
        src/lgc.c
        src/lzio.c
        src/ldblib.c
        src/strbuf.c
        src/lmem.c
        src/lcode.c
        src/ltablib.c
        src/lua_struct.c
        src/lapi.c
        src/lbaselib.c
        src/lua_cmsgpack.c
        src/ldebug.c
        src/lparser.c
        src/lua_cjson.c
        src/fpconv.c
```

```
        src/lua_bit.c
        src/llex.c
        src/ltm.c
        src/ldo.c
        )

add_library(lua STATIC ${LUA_SRC})
```

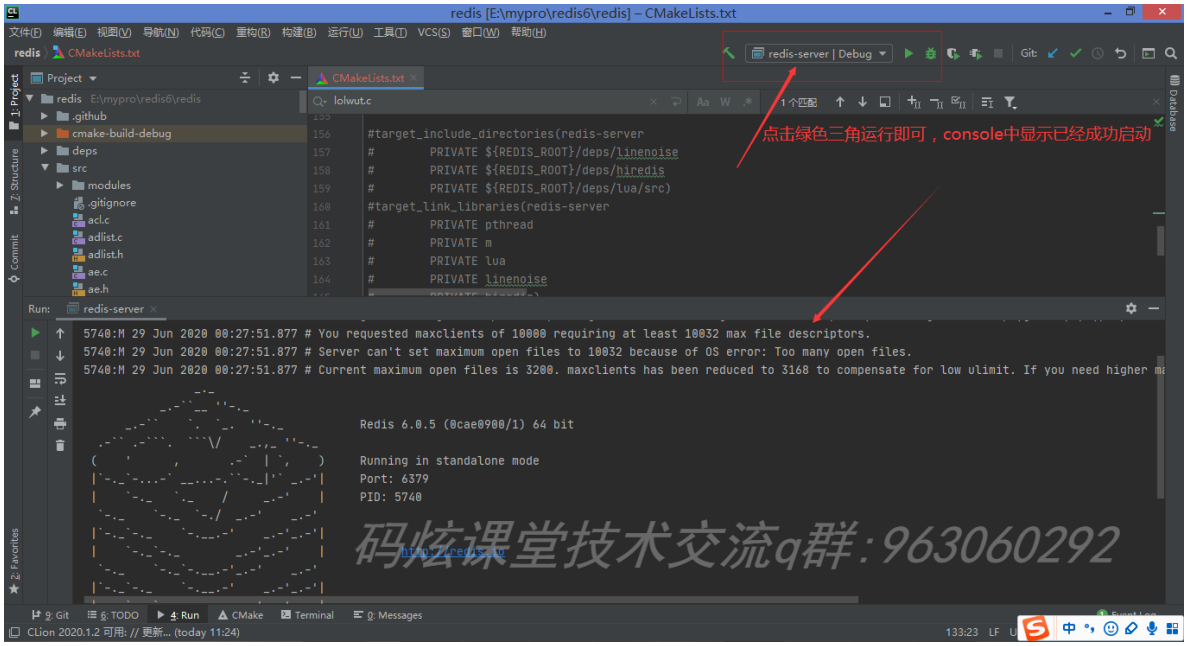
./src/modules/CMakeLists.txt修改如下：

```
cmake_minimum_required(VERSION 3.9)
set(CMAKE_BUILD_TYPE "Debug")
add_library(helloworld SHARED helloworld.c)
set_target_properties(helloworld PROPERTIES PREFIX "" SUFFIX ".so")


add_library(hellotype SHARED hellotype.c)
set_target_properties(hellotype PROPERTIES PREFIX "" SUFFIX ".so")


add_library(helloblock SHARED helloblock.c)
set_target_properties(helloblock PROPERTIES PREFIX "" SUFFIX ".so")


add_library(testmodule SHARED testmodule.c)
set_target_properties(testmodule PROPERTIES PREFIX "" SUFFIX ".so")
```

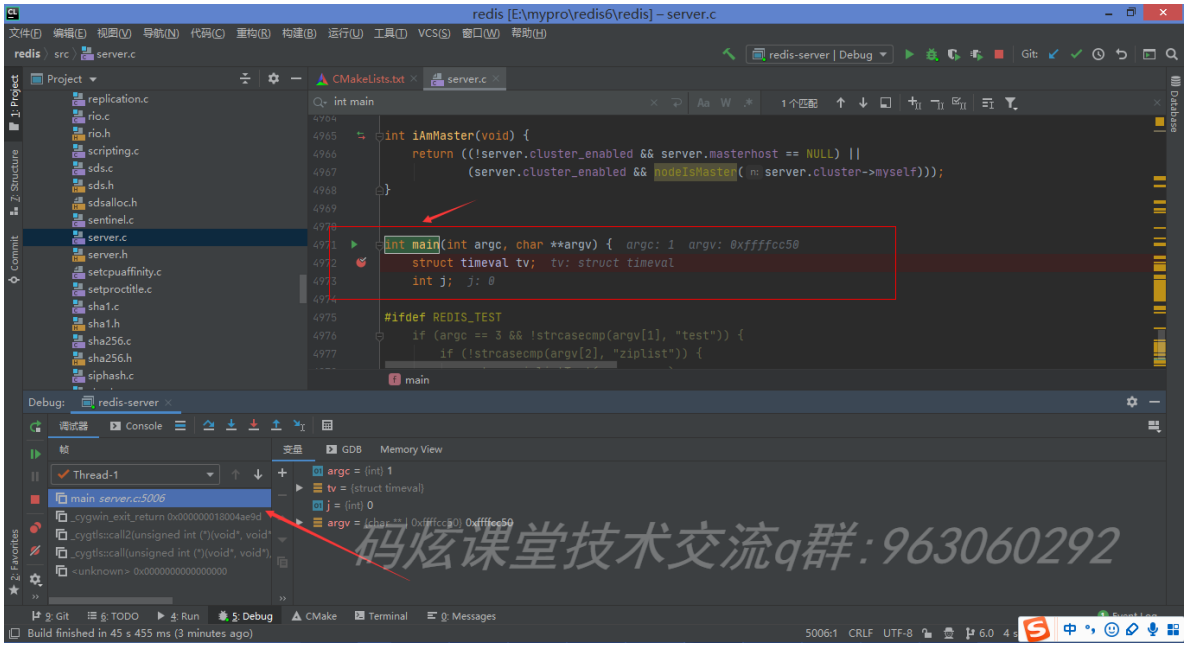
五、编译&调试redis6源码

1）、redis6源码加载之后，cmake会自动编译，编译完成后，选择**redis-server|debug**选项，点击【运行】即可。

撒花，成功！redis版本为当前最新6.0.5版本，如下图所示：



2）、验证是不是真的可以调试，可以打开 `server.c` 文件，在 `int main(int argc, char **argv)` 方法中，添加断点，开始愉快的调试。如下图：



嘿嘿，比想象中的顺利。如果各位同学在搭建调试的过程中，有碰到问题，可以加入**技术交流q群：963060292** 给我留言。smart哥自己在配置 `./CMakeLists.txt` 文件时卡壳了，不同版本的 Redis 会有所不同。

六、注意点

1、在编译之前，需要在Cygwin64 Terminal上执行 `mkreleasehdr.sh` 脚本，由于该脚本格式是win格式，所以需要执行dos2unix命令转换，转换完之后再执行。

```
##转换格式
dos2unix mkreleasehdr.sh

##执行mkreleasehdr.sh
./mkreleasehdr.sh
```

2、运行到最后一步会报如下错误：

error while loading shared libraries: cyghiredis.dll: cannot open shared object file: No such file or directory

解决方案：需要把E:\mypro\redis6\redis\cmake-build-debug\deps\hiredis\cyghiredis.dll拷贝到E:\mypro\redis6\redis\cmake-build-debug\目录下。

3、网上现有redis3版本CMakeLists.txt文件的相关配置照搬到redis6中是无法运行的。

相关CMakeLists.txt文件的具体配置获取请移步至：**码炫课堂java架构师技术交流群：963060292**

smart哥首创**4位1体**打法之--**源码篇**-最新【**redis6源码解析**】课程即将开启，全盘解析redis6中IO多路复用等特性的底层源码，该课程对交流群中的小伙伴限时免费开放！