



中国科学院空天信息研究院

Aerospace Information Research Institute(AIR)
Chinese Academy of Sciences(CAS)

Cmake、Make介绍

汇报人：李顺

2020年7月14日





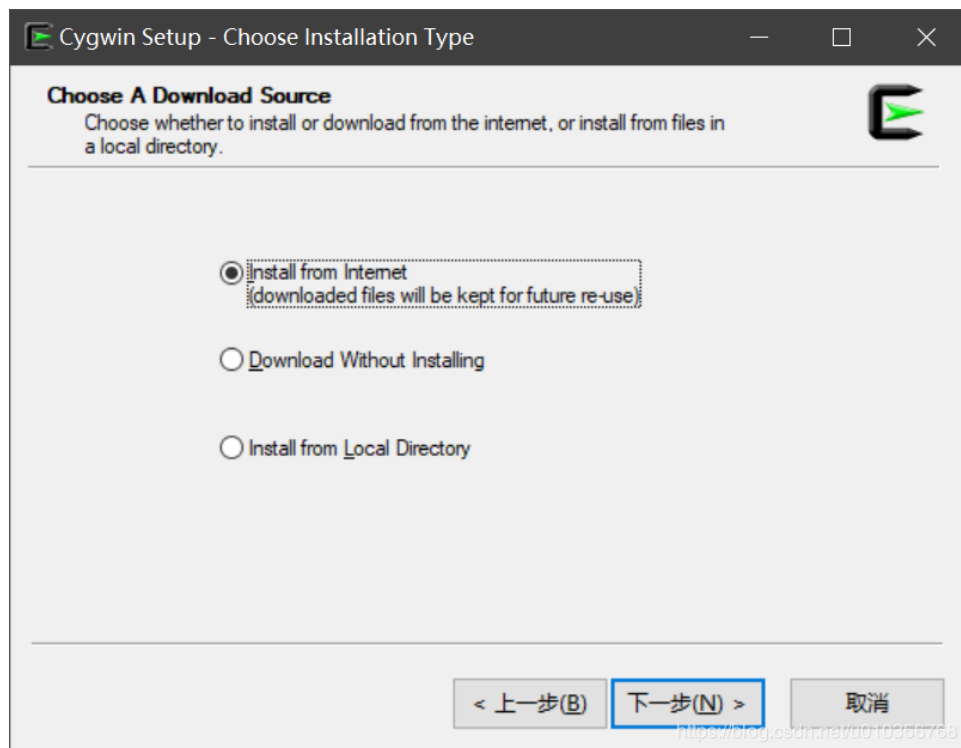
提 纲

- 一 Cygwin安装教程
- 二 用CMake、Make编译成可执行文件
- 三 用Cmake、Make编译静态/动态库
- 四 链接静态/动态库文件生成可执行文件
- 五 第一阶段工作安排



Cygwin安装教程

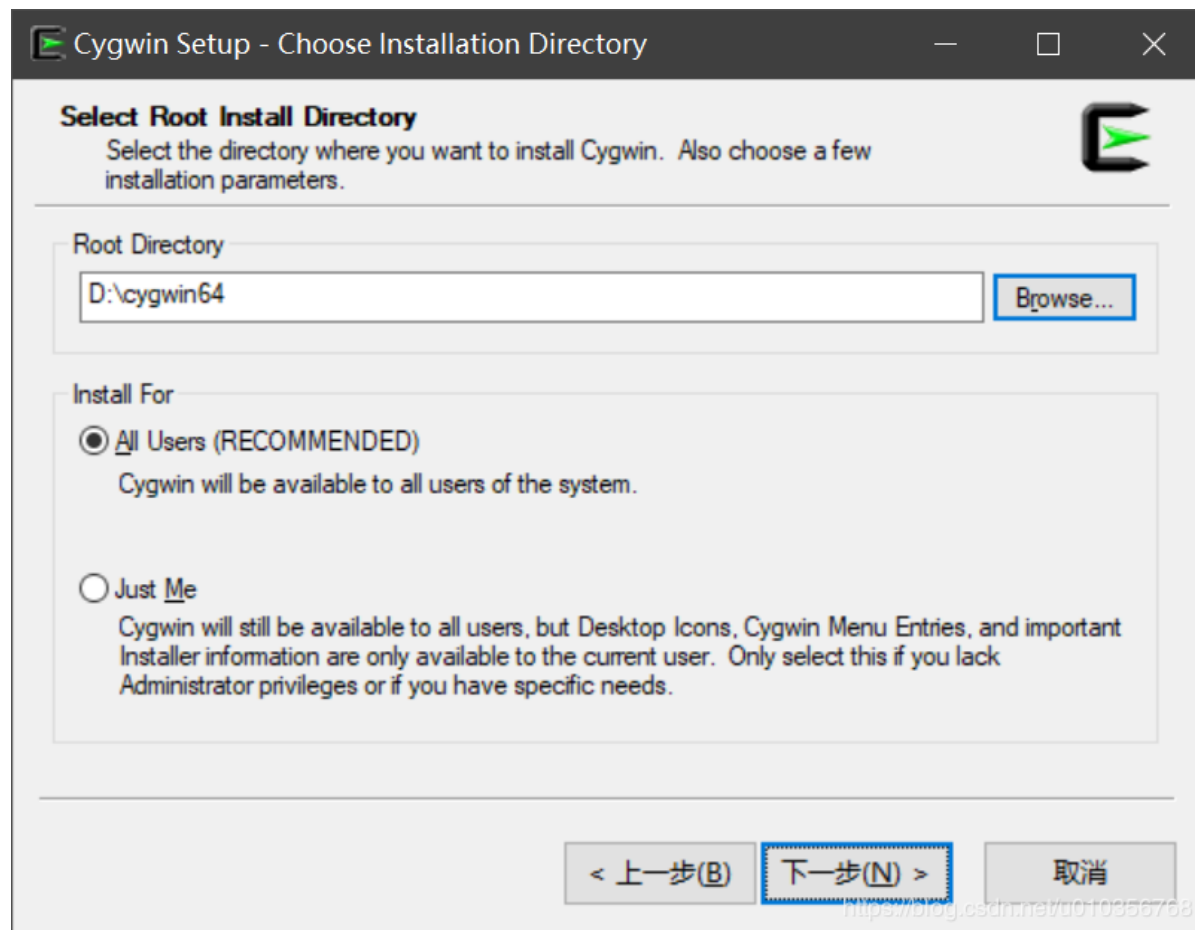
- 1、外网下载地址: <http://www.cygwin.com/> 内网在引导网站的环境配置下的常用工具配置下的Windows下搭建linux模拟环境Cygwin
- 2、双击下载好的setup-x86_64.exe,进入下载页面
- 3、点击下一步,有三种安装模式,一般我们选取第一种, install from Internet





Cygwin安装教程

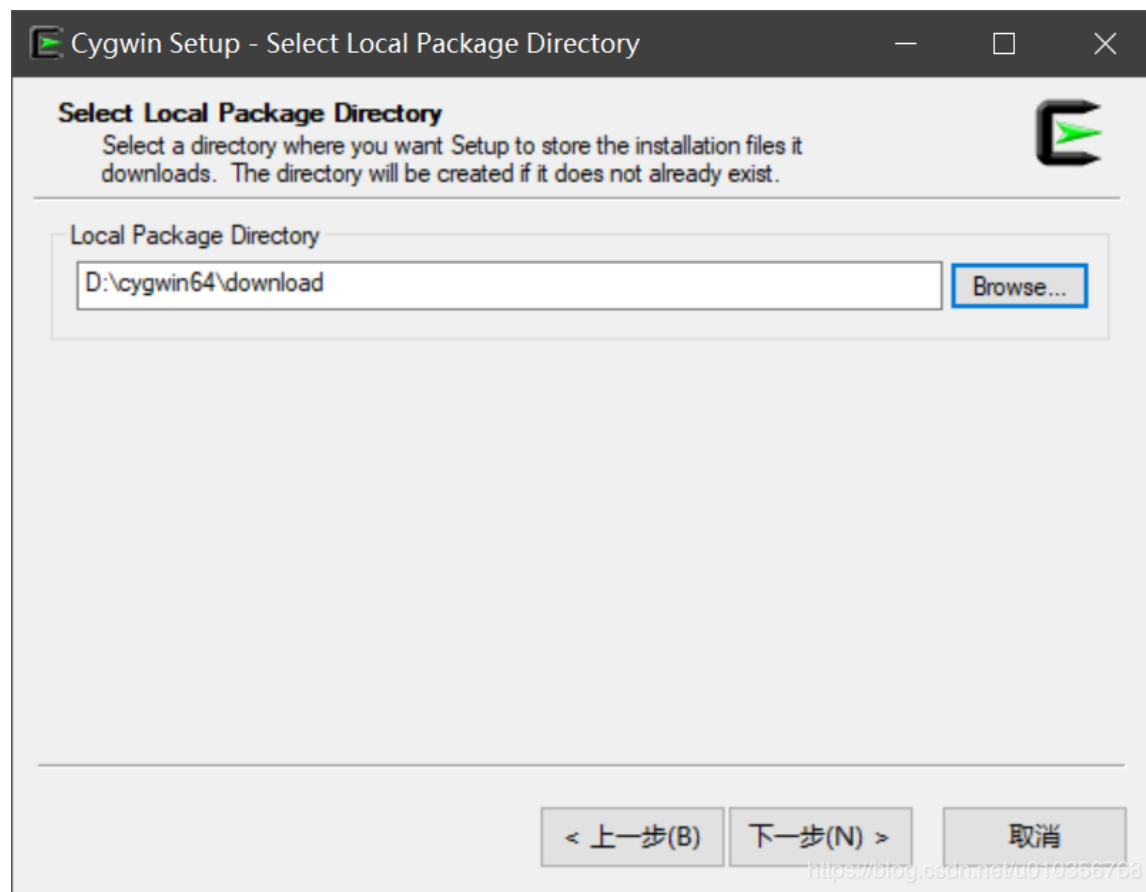
4、点击下一步，选择安装路径





Cygwin安装教程

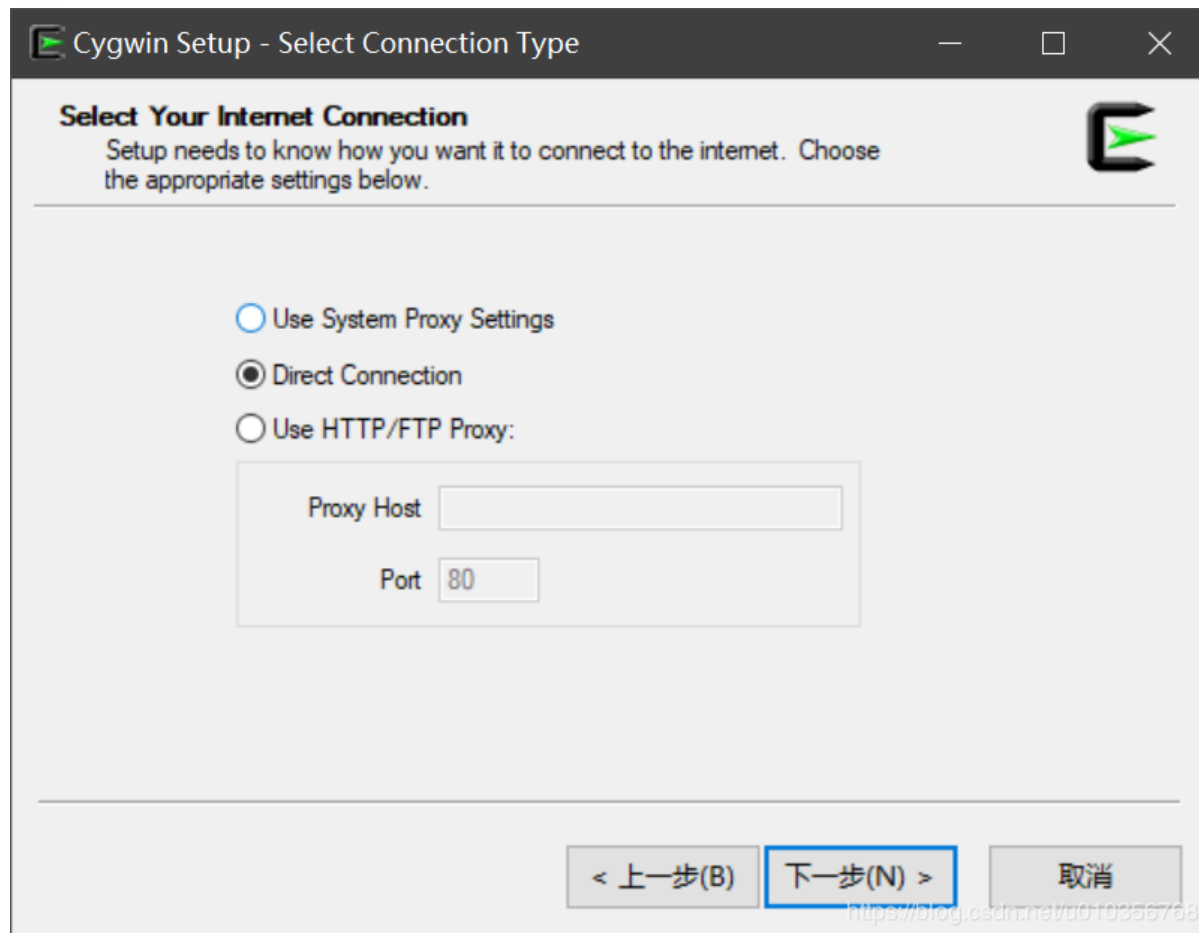
5、点击下一步，在下载的同时，Cygwin组件也保存到了本地，以便以后能够再次安装，这一步选择安装过程中从网上下载的Cygwin组件包的保存位置





Cygwin安装教程

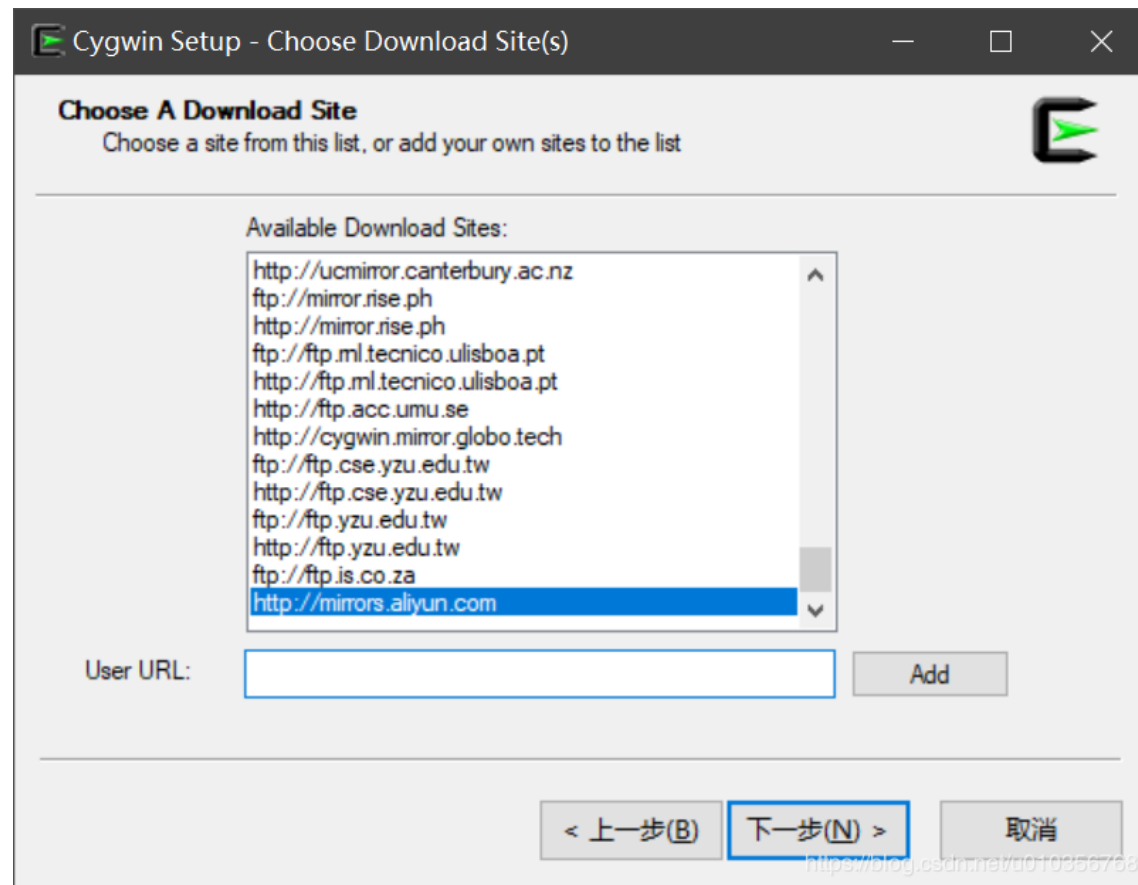
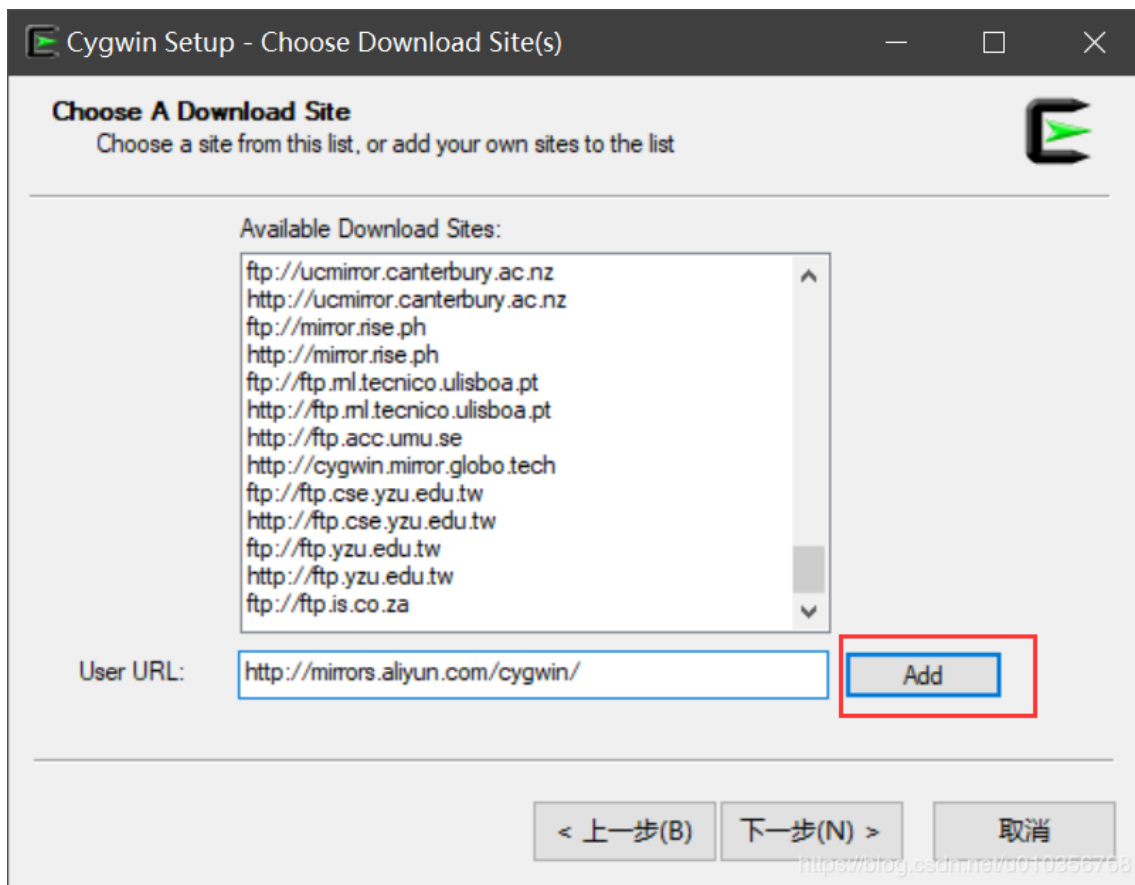
6、点击下一步，选择连接方式，一般选择默认的 Direct Connection





Cygwin安装教程

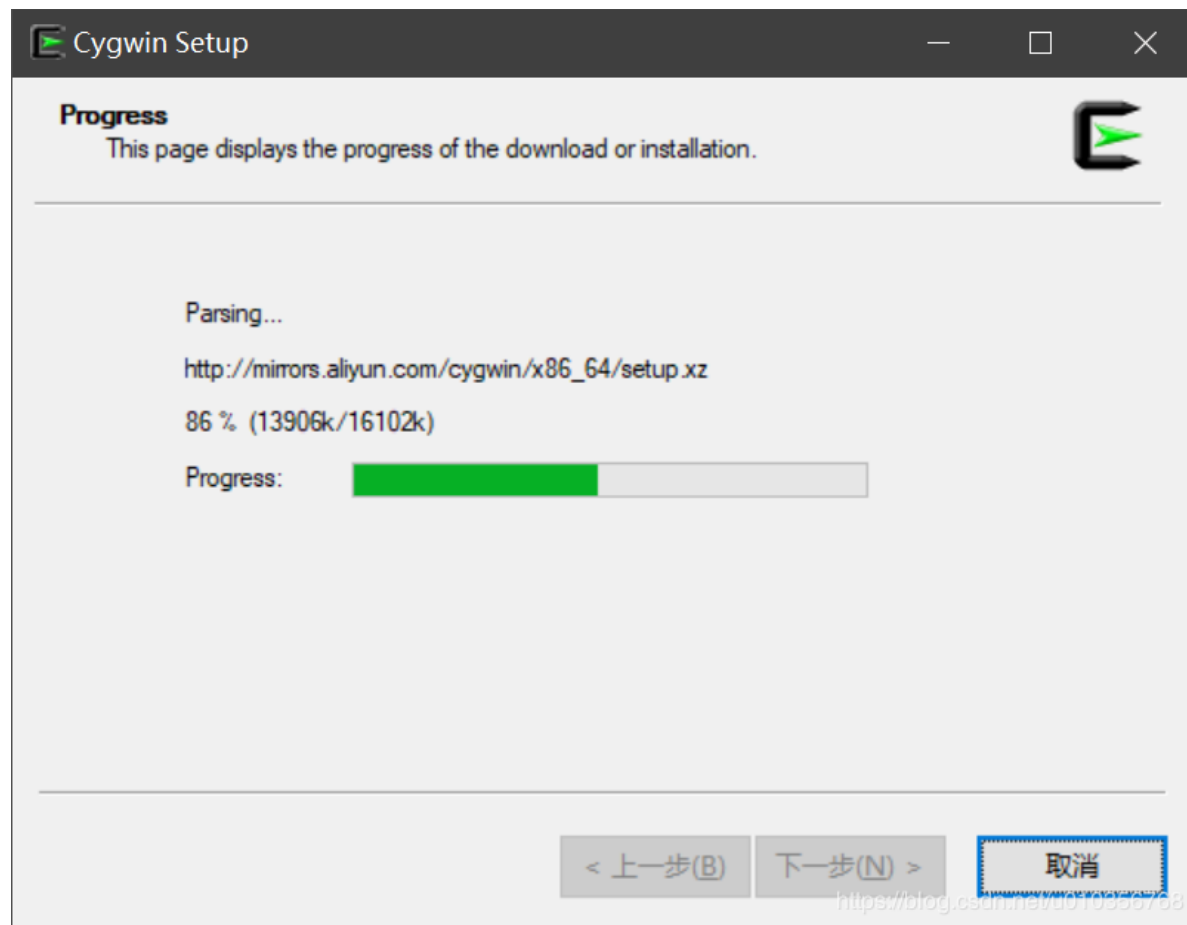
7、点击下一步，选择下载站点，为获得最快的下载速度，我们可以添加网易开源镜像
<http://mirrors.163.com/cygwin/> 或者 阿里云镜像<http://mirrors.aliyun.com/cygwin/>





Cygwin安装教程

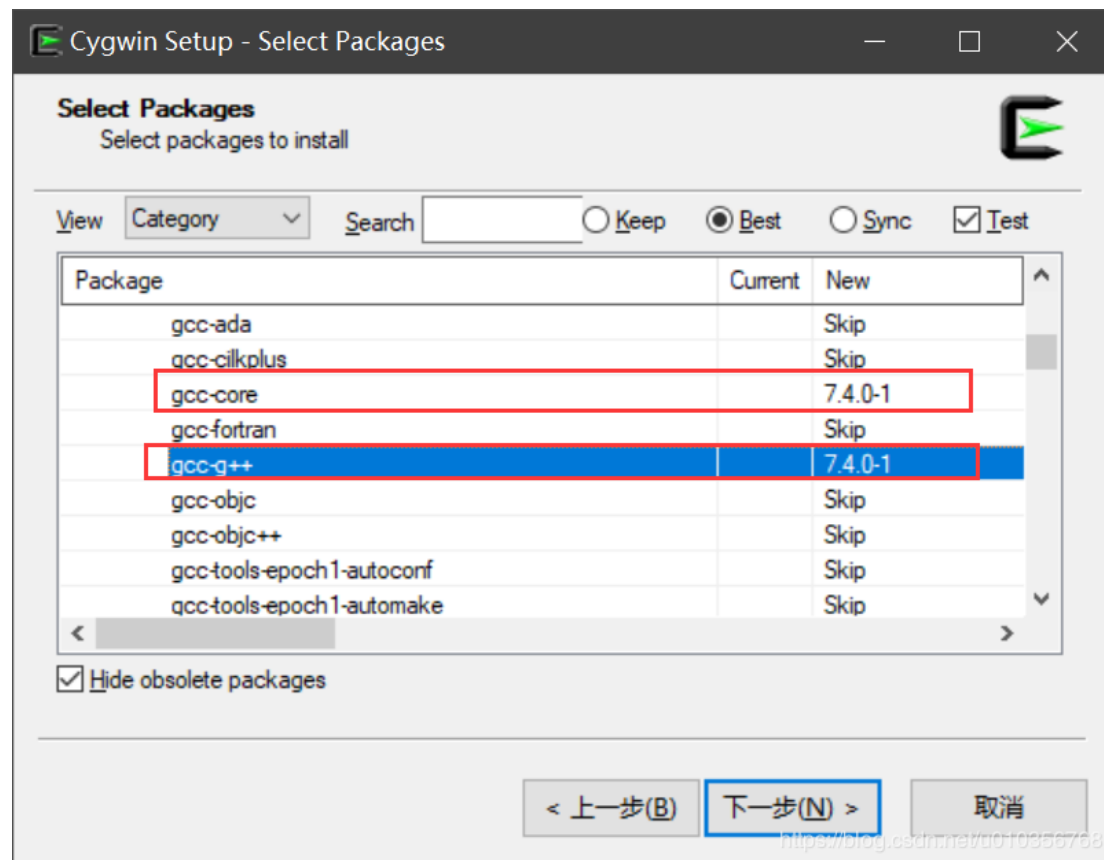
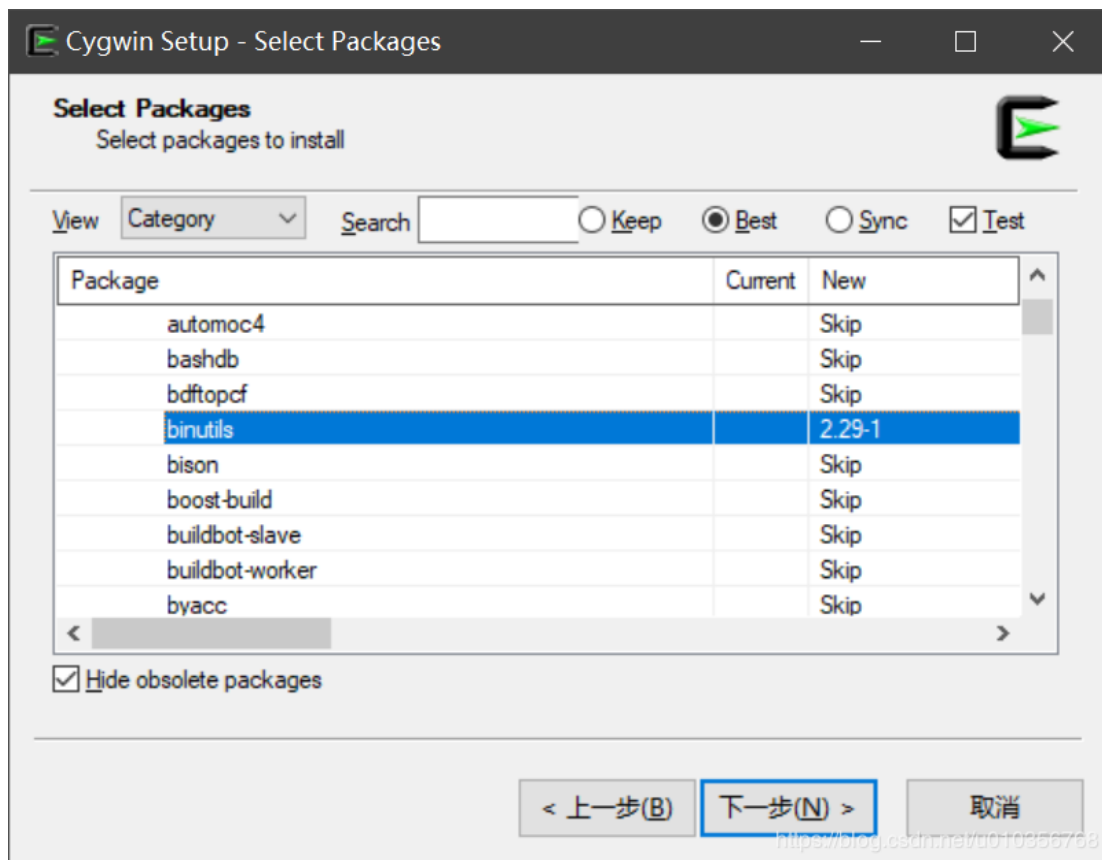
8、点击下一步，组件会开始加载





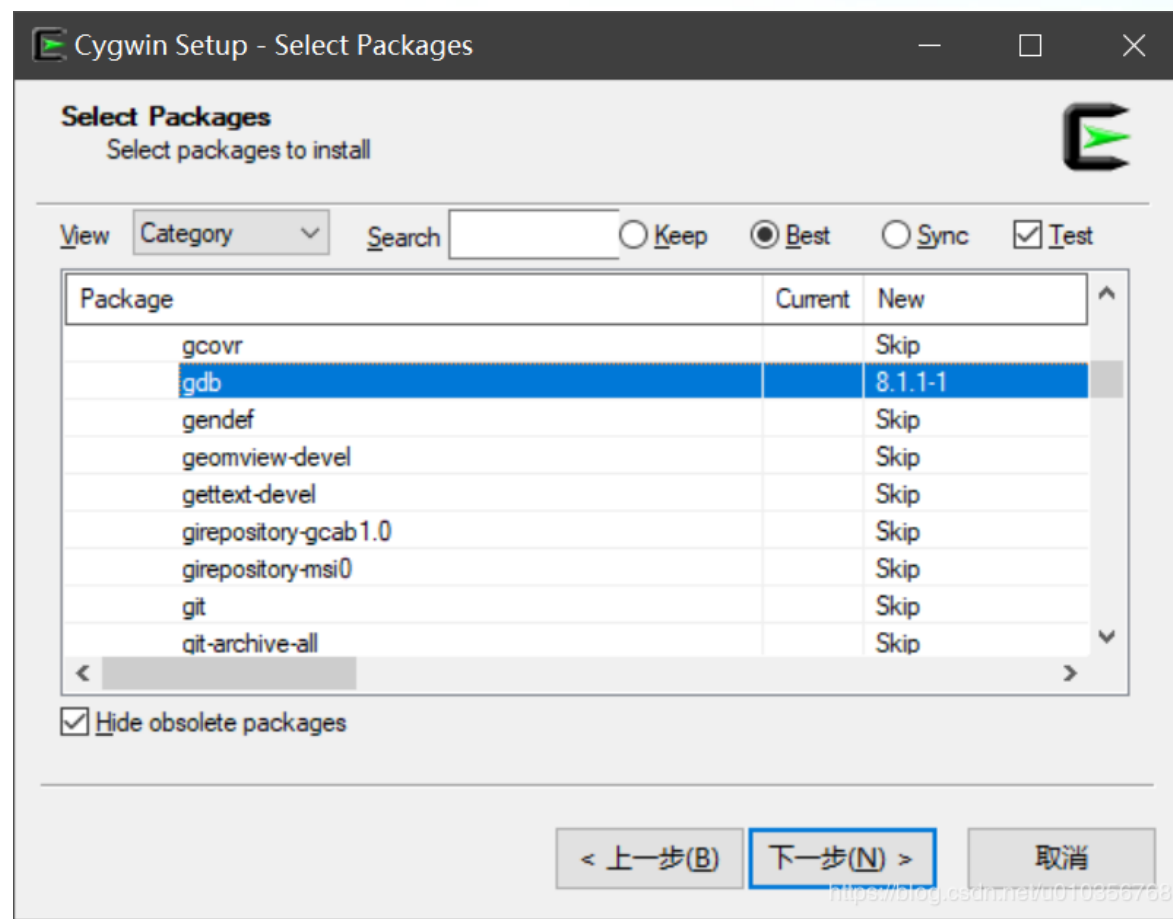
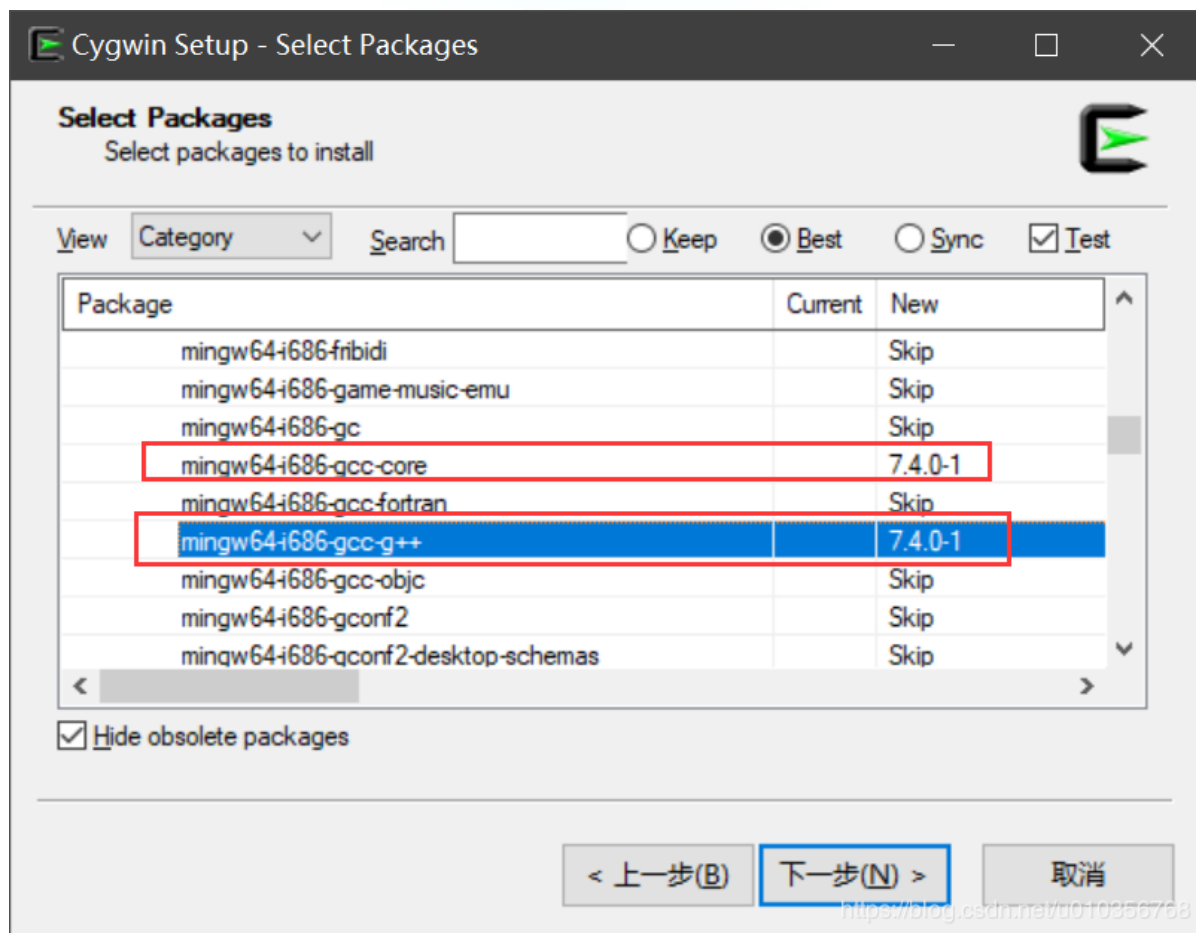
Cygwin安装教程

9、加载完之后选择需要的模块，必须安装的模块有binutils、gcc、mingw、gdb、cmake和make；可以在左上角的搜索框中依次搜索下载。以下放上需要下载的模块截图：



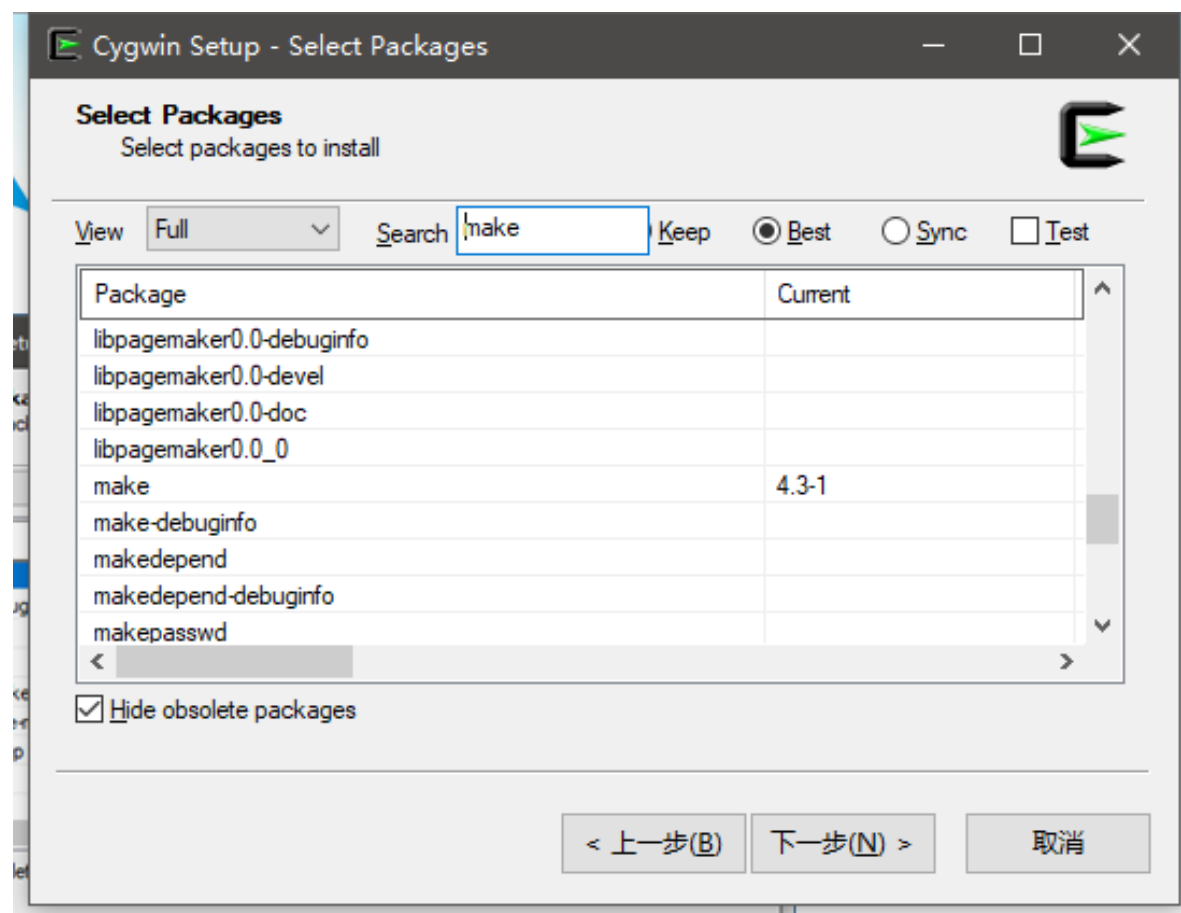
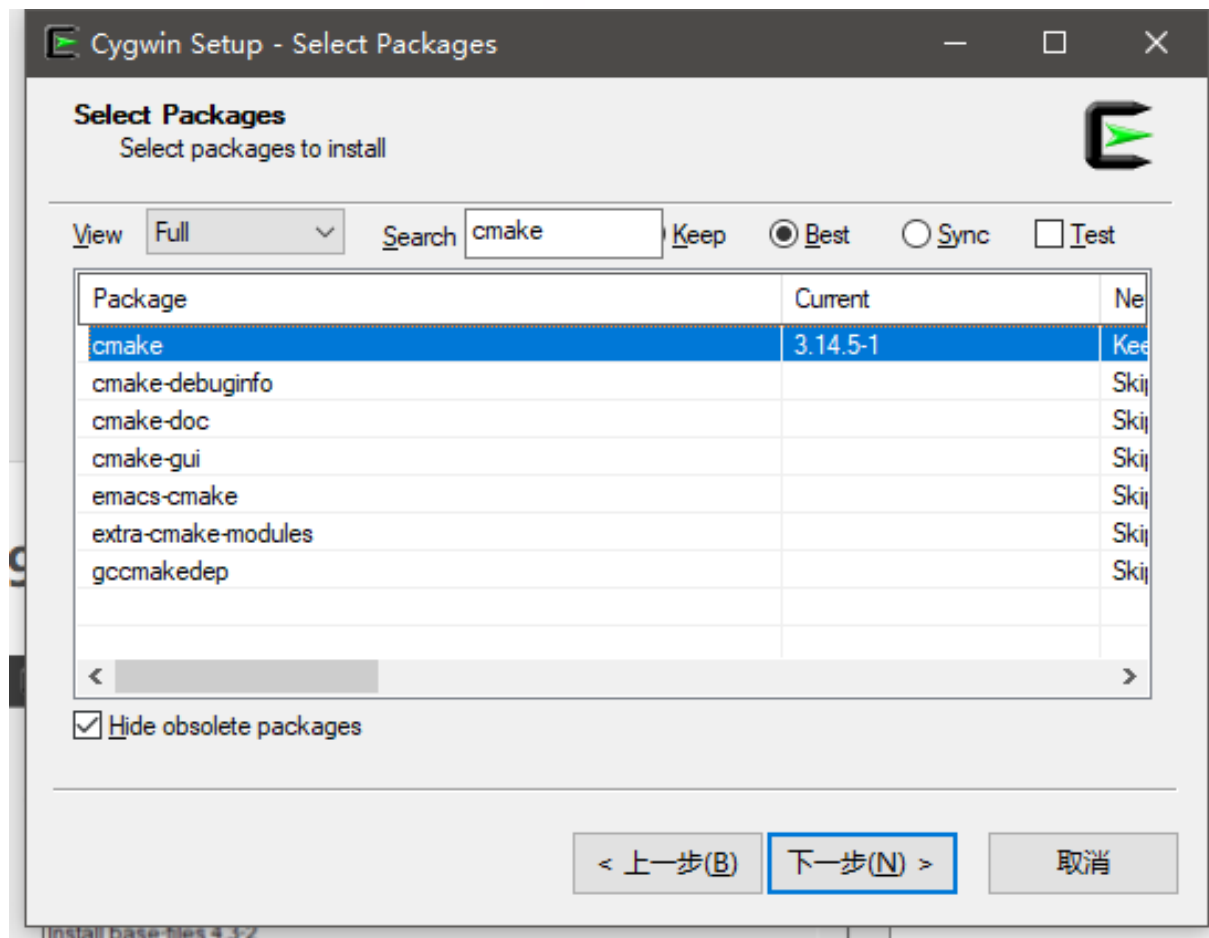


Cygwin安装教程





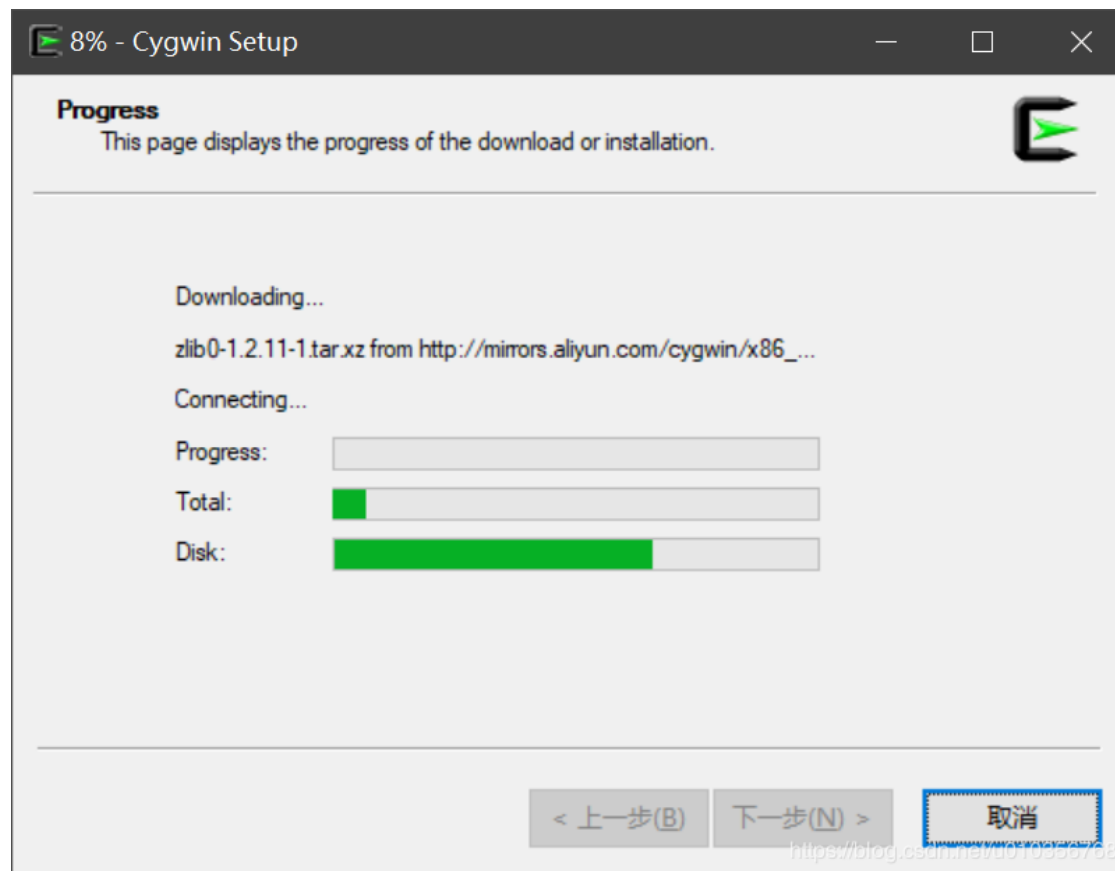
Cygwin安装教程





Cygwin安装教程

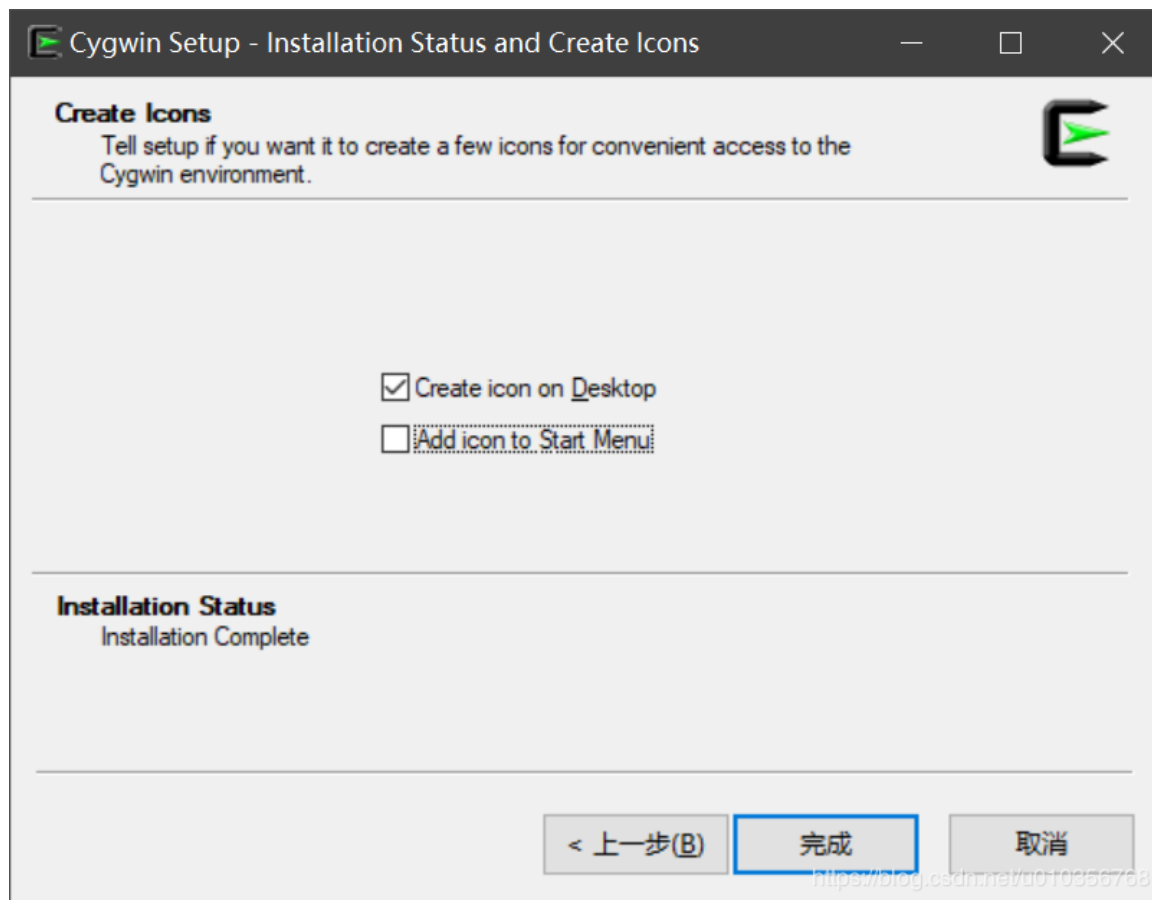
10、选择完后，点击下一步，等待安装完成





Cygwin安装教程

11、安装完成，创建桌面快捷方式





Cygwin安装教程

12、验证cygwin是否安装成功，首先运行cygwin，在弹出的命令窗口输入
`cygcheck -c cygwin`；会打印出当前cygwin的版本和运行状态，如果status是
ok的话，则cygwin运行正常。

```
ASUS-L@DESKTOP-HP9N36R /cygdrive/f/company/g4
$ cygcheck -c cygwin
Cygwin Package Information
Package          Version          Status
cygwin            3.1.5-1          OK
```

13、验证cmake和make是否安装成功，在窗口中分别输入`cmake --version`和
`make --version`；显示出相应的版本即安装成功。到此Cygwin的环境配置完成。



提 纲

- 一 Cygwin安装教程
- 二 用Cmake、Make编译成可执行文件
- 三 用Cmake、Make编译静态/动态库
- 四 链接静态/动态库文件生成可执行文件
- 五 第一阶段工作安排



用CMake、Make编译成可执行文件

1、首先创建一个项目文件夹，在文件夹中编写一个c++文件-----main.cpp

main.cpp内容如下：

```
#include <iostream>

int main ()
]{
    std::cout << "Hello, world!" << std::endl;
    return 0;
}
```



用CMake、Make编译成可执行文件

2、然后编写CMakeLists.txt文件

```
cmake_minimum_required (VERSION 3.2)#指定运行此配置文件所需的CMake的最低版本；

set(PROJECT_NAME main)#设置项目名字变量
project(${PROJECT_NAME})#设置项目名字 ---》项目编译之后生成的可执行文件名字或者静态库/动态库文件名字

set(CMAKE_CXX_FLAGS "${CMAKE_CXX_FLAGS} -g -std=c++11")#在编译C++代码时加上C++11支持选项

file(GLOB cpp_src_file "*.cpp") #查找当前文件夹中所有.cpp结尾的源代码文件
SET(DIR_SRCS ${cpp_src_file})) #将找到的所有文件设置为一个变量

add_executable(${PROJECT_NAME} ${DIR_SRCS})#将源文件编译成可执行文件
```



用Cmake、Make编译成可执行文件

3、创建一个空的文件夹，命名为build；该文件夹用来存放编译之后的文件，避免打乱根目录结构。

4、双击运行下载好的Cygwin，用cd命令切换到自己创建的项目目录下，cd build 进入创建的空文件夹，在该文件夹内运行cmake .. 指令，在出现如下图所示的提示时

```
-- Generating done
-- Build files have been written to: /cygdrive/f/company/基础平台开发/CMakeDemo/build
```

就可以运行 make 指令，完成最终的编译过程。出现如下图 Linking main.exe

```
ASUS-L@DESKTOP-HP9N36R /cygdrive/f/company/基础平台开发/CMakeDemo/build
$ make
make[1]: 进入目录"/cygdrive/f/company/基础平台开发/CMakeDemo/build"
make[2]: 进入目录"/cygdrive/f/company/基础平台开发/CMakeDemo/build"
Scanning dependencies of target main
make[2]: 离开目录"/cygdrive/f/company/基础平台开发/CMakeDemo/build"
make[2]: 进入目录"/cygdrive/f/company/基础平台开发/CMakeDemo/build"
[ 50%] Building CXX object CMakeFiles/main.dir/main.cpp.o
[100%] Linking CXX executable main.exe
make[2]: 离开目录"/cygdrive/f/company/基础平台开发/CMakeDemo/build"
[100%] Built target main
make[1]: 离开目录"/cygdrive/f/company/基础平台开发/CMakeDemo/build"
ASUS-L@DESKTOP-HP9N36R /cygdrive/f/company/基础平台开发/CMakeDemo/build
```




用Cmake、Make编译成可执行文件

5、最后在build目录下运行main.exe文件，最终显示如下所示：

```
ASUS-L@DESKTOP-HP9N36R /cygdrive/f/company/基础平台开发/CMakeDemo/build  
$ ./main  
Hello, world!
```

到此，用Cmake、Make将C++项目编译成可执行文件结束。



提 纲

- 一 Cygwin安装教程
- 二 用CMake、Make编译成可执行文件
- 三 用Cmake、Make编译静态/动态库
- 四 链接静态/动态库文件生成可执行文件
- 五 第一阶段工作安排



用Cmake、Make编译静态/动态库

一、编译成静态库

- 1、新建一个项目文件夹，编写一个头文件MathFunctions.h声明一个函数，MathFunctions.h 的内容如下：

```
#ifndef MATHFUNCTIONS_H          //避免头文件被#include多次
#define MATHFUNCTIONS_H

extern double power(double base, int exponent); //extern用来声明函数的作用范围

#endif
```



用Cmake、Make编译静态/动态库

2、编写一个C++文件MathFunctions.cpp定义这个函数，MathFunctions.cpp内容如下图所示：

```
double power(double base, int exponent)
{
    int result = base;
    int i;

    if (exponent == 0)
    {
        return 1;
    }

    for(i = 1; i < exponent; ++i)
    {
        result = result * base;
    }

    return result;
}
```



用Cmake、Make编译静态/动态库

3、新建一个空文件夹命名为build,新建一个名为output的空文件夹，output文件夹中存放编译生成的静态库文件；编写CMakeLists.txt文件，内容如下：

```
Cmake_minimum_required(VERSION 3.2)

set(PROJECT_NAME demo1)
project(${PROJECT_NAME})

#aux_source_directory(. DIR_SRCS)
file(GLOB cpp_src_file "*.cpp" "*.cc")#在当前文件夹内找到.cpp/.cc结尾的源文件
SET(DIR_SRCS ${cpp_src_file})

ADD_LIBRARY(${PROJECT_NAME} ${DIR_SRCS}) #将源文件编译成一个静态库文件

SET(LIBRARY_OUTPUT_PATH ${PROJECT_SOURCE_DIR}/output/)#将动态库静态库输出到指定的目录
```




用Cmake、Make编译静态/动态库

4、用cd 命令进入到新建的工程目录下的build文件夹，运行cmake ..指令，运行结果如下：

```
Detecting CXX compile features - done
Configuring done
Generating done
Build files have been written to: /cygdrive/f/company/geobos/CMakeDemo5/build
```



即可运行make 指令，显示如下图：

```
ASUS-L@DESKTOP-HP9N36R /cygdrive/f/company/geobos/CMakeDemo5/build
$ make
make[1]: 进入目录"/cygdrive/f/company/geobos/CMakeDemo5/build"
make[2]: 进入目录"/cygdrive/f/company/geobos/CMakeDemo5/build"
Scanning dependencies of target demo1
make[2]: 离开目录"/cygdrive/f/company/geobos/CMakeDemo5/build"
make[2]: 进入目录"/cygdrive/f/company/geobos/CMakeDemo5/build"
[ 50%] Building CXX object CMakeFiles/demo1.dir/MathFunctions.cpp.o
[100%] Linking CXX static library ../output/libdemo1.a
make[2]: 离开目录"/cygdrive/f/company/geobos/CMakeDemo5/build"
[100%] Built target demo1
make[1]: 离开目录"/cygdrive/f/company/geobos/CMakeDemo5/build"
```



用Cmake、 Make编译静态/动态库

5、出现如上所示结果，可查看output文件夹是否生成以.a结尾的文件，若有则编译静态库成功。

	libdemo1.a	2020/7/10 10:55	A 文件	2 KB
	libdemo1.dll.a	2020/7/10 17:16	A 文件	2 KB



用Cmake、Make编译静态/动态库

二、编译成动态库

1、同样以编译静态库的项目为例，将build文件夹下的东西删除，打开CMakeLists.txt文件，按照以下形式修改：

将 `ADD_LIBRARY(${PROJECT_NAME} ${DIR_SRCS})` 修改成

`ADD_LIBRARY(${PROJECT_NAME} SHARED ${DIR_SRCS})`

#加一个SHARED 表明将源文件编译成动态库



用Cmake、Make编译静态/动态库

2、用cd 命令进入到工程目录下的build文件夹，运行cmake ..指令，运行结果如下：

```
-- Detecting CXX compile features - done
-- Configuring done
-- Generating done
-- Build files have been written to: /cygdrive/f/company/geobos/CMakeDemo5/build
```

查看build文件夹内有没有Makefile文件，有则运行make指令，结果如下所示：

```
ASUS-L@DESKTOP-HP9N36R /cygdrive/f/company/geobos/CMakeDemo5/build
$ make
make[1]: 进入目录"/cygdrive/f/company/geobos/CMakeDemo5/build"
make[2]: 进入目录"/cygdrive/f/company/geobos/CMakeDemo5/build"
Scanning dependencies of target demo1
make[2]: 离开目录"/cygdrive/f/company/geobos/CMakeDemo5/build"
make[2]: 进入目录"/cygdrive/f/company/geobos/CMakeDemo5/build"
[ 50%] Building CXX object CMakeFiles/demo1.dir/MathFunctions.cpp.o
[100%] Linking CXX shared library ../output/cygdemo1.dll
make[2]: 离开目录"/cygdrive/f/company/geobos/CMakeDemo5/build"
[100%] Built target demo1
make[1]: 离开目录"/cygdrive/f/company/geobos/CMakeDemo5/build"

ASUS-L@DESKTOP-HP9N36R /cygdrive/f/company/geobos/CMakeDemo5/build
```



用Cmake、Make编译静态/动态库

3、查看output文件夹，是否有.dll结尾的文件（windows系统下生成的）

名称	修改日期	类型	大小
 cygdemo1.dll	2020/7/10 11:20	应用程序扩展	139 KB
 libdemo1.a	2020/7/10 10:55	A 文件	2 KB
 libdemo1.dll.a	2020/7/10 11:20	A 文件	3 KB

出现上图所示文件，则动态库编译成功。



提 纲

- 一 Cygwin安装教程
- 二 用CMake、Make编译成可执行文件
- 三 用Cmake、Make编译静态/动态库
- 四 链接静态/动态库文件生成可执行文件
- 五 第一阶段工作安排



链接静态/动态库文件生成可执行文件

一、链接静态库生成可执行文件



1、新建一个工程目录文件夹，编写一个c++文件main.cc，main.cc内容如下：

```
> company > geobos > CMakeDemo7 > main.cc
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include "MathFunctions.h"
4
5  int main(int argc, char *argv[]) //int argc, char *argv[]用于 命令行编译程序中  argc 表示输入的字符个数，argv表示字符
6  {
7      if (argc < 3)
8      {
9          printf("Usage: %s base exponent %d\n", argv[0],argc);
10         return 1;
11     }
12
13     double base = atof(argv[1]);
14     int exponent = atoi(argv[2]);
15     double result = power(base, exponent);
16
17     printf("%g ^ %d is %g\n", base, exponent, result);
18     return 0;
19 }
20
```



链接静态/动态库文件生成可执行文件

2、创建一个空文件夹，命名为thirdparty,里面保存源代码所需要的头文件和第三方库，为了方便管理，再创建一个名为math的文件夹来保存这些头文件和第三方库文件（静态库和动态库）

名称	日期/时间	类型
 libdemo1.a	2020/7/9 23:47	A 文件
 MathFunctions.h	2020/7/8 23:04	H 文件



链接静态/动态库文件生成可执行文件

3、编写CMakeLists.txt文件，内容如下：

```
Cmake_minimum_required(VERSION 3.2)

set(PROJECT_NAME test)
project(${PROJECT_NAME})

set(INC_DIR ./thirdparty/math) # 设置第三方头文件的路径
set(LIB_DIR ./thirdparty/math) # 设置依赖库的路径
INCLUDE_DIRECTORIES(${INC_DIR})
LINK_DIRECTORIES(${LIB_DIR})
file(GLOB cpp_src_file "*.cpp" "*.cc")

SET(DIR_SRCS ${cpp_src_file})

add_executable(${PROJECT_NAME} ${DIR_SRCS})

target_link_libraries(${PROJECT_NAME} libdemo1.a)
```



链接静态/动态库文件生成可执行文件

4、新建一个空白的build文件夹，在cygwin界面用cd 命令进入，执行cmake .. 指令，运行截图如下：

```
ASUS-L@DESKTOP-HP9N36R /cygdrive/f/company/geobos/CMakeDemo7/build
$ cmake ..
-- The C compiler identification is GNU 9.3.0
-- The CXX compiler identification is GNU 9.3.0
-- Check for working C compiler: /usr/bin/cc
-- Check for working C compiler: /usr/bin/cc -- works
-- Detecting C compiler ABI info
-- Detecting C compiler ABI info - done
-- Detecting C compile features
-- Detecting C compile features - done
-- Check for working CXX compiler: /usr/bin/c++.exe
-- Check for working CXX compiler: /usr/bin/c++.exe -- works
-- Detecting CXX compiler ABI info
-- Detecting CXX compiler ABI info - done
-- Detecting CXX compile features
-- Detecting CXX compile features - done
-- Configuring done
-- Generating done
-- Build files have been written to: /cygdrive/f/company/geobos/CMakeDemo7/build
```




链接静态/动态库文件生成可执行文件

5、执行make 指令，运行结果如下：

```
ASUS-L@DESKTOP-HP9N36R /cygdrive/f/company/geobos/CMakeDemo7/build
$ make
make[1]: 进入目录"/cygdrive/f/company/geobos/CMakeDemo7/build"
make[2]: 进入目录"/cygdrive/f/company/geobos/CMakeDemo7/build"
Scanning dependencies of target test
make[2]: 离开目录"/cygdrive/f/company/geobos/CMakeDemo7/build"
make[2]: 进入目录"/cygdrive/f/company/geobos/CMakeDemo7/build"
[ 50%] Building CXX object CMakeFiles/test.dir/main.cc.o
[100%] Linking CXX executable test.exe
make[2]: 离开目录"/cygdrive/f/company/geobos/CMakeDemo7/build"
[100%] Built target test
make[1]: 离开目录"/cygdrive/f/company/geobos/CMakeDemo7/build"
```



链接静态/动态库文件生成可执行文件

6、运行exe执行文件，运行结果如下：

```
ASUS-L@DESKTOP-HP9N36R /cygdrive/f/company/geobos/CMakeDemo7/build  
$ ./test 4 5  
4 ^ 5 is 1024
```



到此链接静态库成功。



链接静态/动态库文件生成可执行文件

二、链接动态库生成可执行文件

1、保持之前链接静态库的目录结构不变，在thirdparty文件夹下的math文件夹内添加一个我们之前编译生成的动态库文件-----cygdemo1.dll

 cygdemo1.dll	2020/7/10 11:20	应用程序扩展	139 K
 MathFunctions.h	2020/7/8 23:04	H 文件	1 K



链接静态/动态库文件生成可执行文件

2、修改CMakeLists.txt文件如下：

```
Cmake_minimum_required(VERSION 3.2)

set(PROJECT_NAME test)
project(${PROJECT_NAME})

set(INC_DIR ./thirdparty/math) # 设置第三方头文件的路径
set(LIB_DIR ./thirdparty/math) # 设置依赖库的路径
INCLUDE_DIRECTORIES(${INC_DIR})
LINK_DIRECTORIES(${LIB_DIR})
file(GLOB cpp_src_file "*.cpp" "*.cc")

SET(DIR_SRCS ${cpp_src_file})

add_executable(${PROJECT_NAME} ${DIR_SRCS})

target_link_libraries(${PROJECT_NAME} cygdemo1.dll)

SET(EXECUTABLE_OUTPUT_PATH ../thirdparty/math)#指定生成的可执行文件所放的目录
```



链接静态/动态库文件生成可执行文件

3、在cygwin界面用cd命令进入build文件夹，执行cmake ..指令，查看build文件夹内是否有Makefile文件，有则使用make 指令，最后结果如下：

```
ASUS-L@DESKTOP-HP9N36R /cygdrive/f/company/geobos/CMakeDemo7/build
$ make
make[1]: 进入目录"/cygdrive/f/company/geobos/CMakeDemo7/build"
make[2]: 进入目录"/cygdrive/f/company/geobos/CMakeDemo7/build"
Scanning dependencies of target test
make[2]: 离开目录"/cygdrive/f/company/geobos/CMakeDemo7/build"
make[2]: 进入目录"/cygdrive/f/company/geobos/CMakeDemo7/build"
[ 50%] Building CXX object CMakeFiles/test.dir/main.cc.o
[100%] Linking CXX executable ../thirdparty/math/test.exe
make[2]: 离开目录"/cygdrive/f/company/geobos/CMakeDemo7/build"
[100%] Built target test
make[1]: 离开目录"/cygdrive/f/company/geobos/CMakeDemo7/build"
```




链接静态/动态库文件生成可执行文件

4、最后进入到生成的可执行文件的文件夹thirdparty/math下，运行可执行文件：

```
ASUS-L@DESKTOP-HP9N36R /cygdrive/f/company/geobos/CMakeDemo7
$ cd thirdparty/math

ASUS-L@DESKTOP-HP9N36R /cygdrive/f/company/geobos/CMakeDemo7/thirdparty/math
$ ./test 4 5
4 ^ 5 is 1024
```



提 纲

- 一 Cygwin安装教程
- 二 用CMake、Make编译成可执行文件
- 三 用Cmake、Make编译静态/动态库
- 四 链接静态/动态库文件生成可执行文件
- 五 第一阶段工作安排



第一阶段工作安排

完成基础环境配置，完成Cmake、Make环境配置、完成读写文件的Demo

- + build
- + output
 - main.exe
- + thirdparty
 - libfilesystem.a
 - file_system.h
- + CMakeLists.txt
- + main.cpp