Windows系统下Redis的安装

自力更生版

参考博客

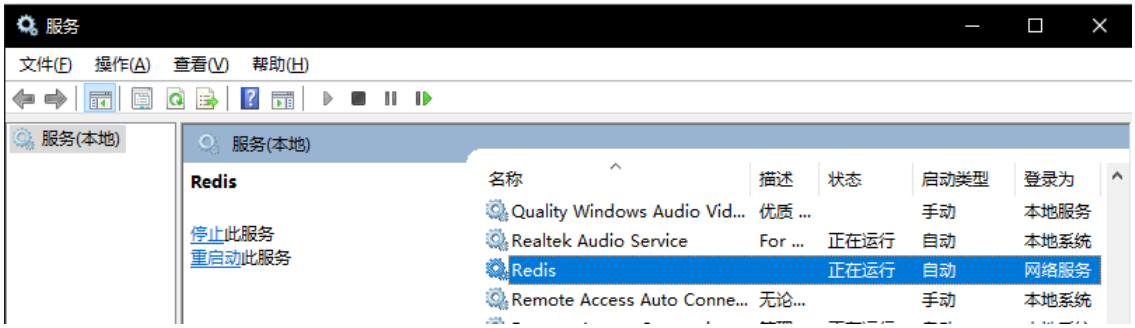
> 注：该博客中的下载地址已停止更新，版本较老，建议从此处下载

以下为凑字数部分

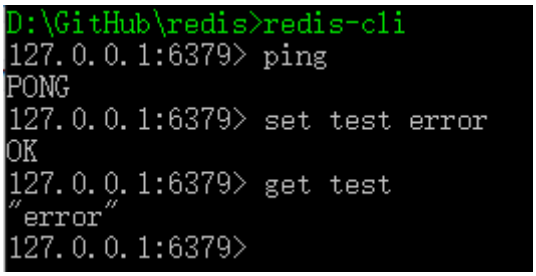
Redis下载地址

> 该链接和自力更生版中的建议下载链接为同一个，下载msi或者zip都行
>
> 使用msi安装起来会比较简单，但因为太过简单所以此处不讲，建议自力更生

1. 下载zip，解压到某路径下，以 `D:\GitHub\redis` 为例
2. 打开cmd，并进入 `D:\GitHub\redis`
3. 运行 `redis-server.exe --service-install redis.windows.conf` ，成功后会得到 `Redis successfully installed as a service` 的提示
4. 运行 `redis-server --service-start` 打开服务，也可以在服务（在任务栏搜索框输入“服务”即可打开该应用）中查看并设置为自动开启



5. 运行 `redis-cli` 连接服务并进行测试



6. 可选，将Redis路径加入系统环境变量。在任务栏搜索框输入“环境变量”，在系统变量的 `Path` 末尾加上Redis的解压路径，示例中为 `D:\GitHub\redis`
   如果选择不修改环境变量，则每次打开cmd后，连接Redis服务前都需要先进入解压路径

Windows系统下使用hiredis

hiredis作为官方的C语言客户端，其实并不支持Windows。而微软为了助力Redis的发展（分一杯羹），也推出了对应的Windows版本（其实就是套了层皮），但这个东西配置起来可能有点麻烦，请耐心

hiredis在windows下的编译以及使用

- 配置过程请参考上方链接的博客

- 博客中几处配图出现了 `win32_fixes.c` 和 `win32_fixes.c` 两份文件（如图），但其实不是必要的，包括代码中也不需要对该头文件进行include

在自己的项目中使用hiredis

新建项目

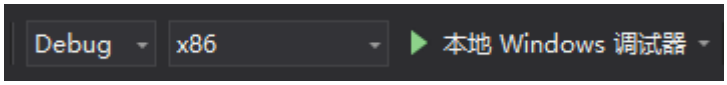
新建项目，项目结构如下所示（其中Debug\、x64\文件夹是生成的，可忽略）：

名称	修改日期	类型	大小
Debug	2019/12/3 16:23	文件夹	
include	2019/12/3 16:23	文件夹	
lib	2019/12/3 16:11	文件夹	
x64	2019/12/3 16:29	文件夹	
call_redis.vcxproj	2019/12/3 16:56	VC++ Project	8 KB
call_redis.vcxproj.filters	2019/12/3 16:29	VC++ Project Fil...	2 KB
main.cpp	2019/12/3 16:52	C++ Source	3 KB
win32_fixes.c	2016/6/22 18:07	C Source	2 KB
win32_fixes.h	2019/12/3 16:34	C/C++ Header	2 KB

- 博客中提到了需要拷贝.h文件到项目include文件夹中，这里最好把.c文件也一起拷贝过来

> lib文件夹包含上一步得到的.lib文件和.pdb文件。include文件夹包含hiredis\，win32_interop\和adapters\三个文件夹，分别对应hiredis-master\，msvs\win32_interop\和hiredis\adapters\下所有的.h文件。

- 对于项目的设置，还有一点需要注意的是请将解决方案平台设置为 `x86`



- 同时，我将我自己配好的项目打包传到了北航云盘，大家在遇到困难时可以下载下来作为参考，下载链接。注意，因为我在实验时使用的是vs2019，不同版本的vs在打开时可能会出现兼容性问题，请自行解决。
- 另，Windows下的hiredis语法与mac os、Linux下存在微小差异（还不确定造成差异的原因是什么），详情参考下方的示例代码（差异主要在执行命令那块）

```
// hiredistest2.cpp : 此文件包含 "main" 函数。程序执行将在此处开始并结束。
//

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "include\hiredis\hiredis.h"

int main(int argc, char** argv) {
    // 连接Redis，默认IP和端口
    redisContext* context = redisConnect("127.0.0.1", 6379);
    // 检查是否连接成功
    if (!context || context->err) {
        if (context) {
            printf("Error: %s\n", context->errstr);
        }
        else {
            printf("Can't allocate redis context\n");
        }
    }
    // 执行命令：SET Fracture_Ray Sakuzyo
```

```
    redisReply* reply = (redisReply*)redisCommand(context, "SET Fracture_Ray
Sakuzyo");
    // 及时释放，避免内存泄漏
    freeReplyObject(reply);
    // 执行命令：GET Fracture_Ray
    reply = (redisReply*)redisCommand(context, "GET Fracture_Ray");
    // 输出结果字符串
    printf("%s\n", reply->str);
    freeReplyObject(reply);
    // 断开连接
    redisFree(context);
    return 0;
}
```