C++输入输出重定向（3种方法）

《C++输入流和输出流》一节提到，cout 和 cerr、clog 的一个区别是，cout 允许被重定向，而 cerr 和 clog 都不支持。值得一提的是，cin 也允许被重定向。

那么，什么是重定向呢？在默认情况下，cin 只能接收从键盘输入的数据，cout 也只能将数据输出到屏幕上。但通过重定向，cin 可以将指定文件作为输入源，即接收文件中早已准备好的数据，同样 cout 可以将原本要输出到屏幕上的数据转而写到指定文件中。

C++ 中，实现重定向的常用方式有 3 种，本节将一一做详细讲解。

C++ freopen()函数实现重定向

freopen() 定义在 <stdio.h> 头文件中，是 C 语言标准库中的函数，专门用于重定向输入流（包括 scanf()、gets() 等）和输出流（包括 printf()、puts() 等）。值得一提的是，该函数也可以对 C++ 中的 cin 和 cout 进行重定向。

举个例子：

```
#include <iostream>    //cin、cout
#include <string>      //string
#include <stdio.h>     //freopen
using namespace std;
int main()
{
    string name, url;
    //将标准输入流重定向到 in.txt 文件
    freopen("in.txt", "r", stdin);
    cin >> name >> url;

    //将标准输出重定向到 out.txt文件
    freopen("out.txt", "w", stdout);
    cout << name << "\n" << url;
    return 0;
}
```

执行此程序之前，我们需要找到当前程序文件所在的目录，并手动创建一个 in.txt 文件，其包含的内容如下：

```
C++
http://c.biancheng.net/cplus/
```

创建好 in.txt 文件之后，可以执行此程序，其执行结果为：

```
<--控制台中，既不需要手动输入，也没有任何输出
```

与此同时，in.txt 文件所在目录下会自动生成一个 out.txt 文件，其包含的内容和 in.txt 文件相同：

```
C++
http://c.biancheng.net/cplus/
```

显然，通过 2 次调用 freopen() 函数，分别对输入流和输出流重定向，使得 cin 不再接收由键盘输入的数据，而是直接从 in.txt 文件中获取数据；同样，cout 也不再将数据输出到屏幕上，而是写入到 out.txt 文件中。

C++ rdbuf()函数实现重定向

rdbuf() 函数定义在 <ios> 头文件中，专门用于实现 C++ 输入输出流的重定向。

值得一提的是，ios 作为 istream 和 ostream 类的基类，rdbuf() 函数也被继承，因此 cin 和 cout 可以直接调用该函数实现重定向。

rdbuf() 函数的语法格式有 2 种，分别为：

```
streambuf * rdbuf() const;
streambuf * rdbuf(streambuf * sb);
```

> streambuf 是 C++ 标准库中用于表示缓冲区的类，该类的指针对象用于代指某个具体的流缓冲区。

其中，第一种语法格式仅是返回一个指向当前流缓冲区的指针；第二种语法格式用于将 sb 指向的缓冲区设置为当前流的新缓冲区，并返回一个指向旧缓冲区的对象。

举个例子：

```
#include <iostream>
#include <fstream>
using namespace std;
int main()
{
    //打开 in.txt 文件，等待读取
    ifstream fin("in.txt");
    //打开 out.txt 文件，等待写入
    ofstream fout("out.txt");
    streambuf *oldcin;
    streambuf *oldcout;
    char a[100];
    //用 rdbuf() 重新定向，返回旧输入流缓冲区指针
    oldcin = cin.rdbuf(fin.rdbuf());
    //从input.txt文件读入
    cin >> a;
    //用 rdbuf() 重新定向，返回旧输出流缓冲区指针
    oldcout = cout.rdbuf(fout.rdbuf());
    //写入 out.txt
    cout << a << endl;

    //还原标准输入输出流
    cin.rdbuf(oldcin); // 恢复键盘输入
    cout.rdbuf(oldcout); //恢复屏幕输出
    //打开的文件，最终需要手动关闭
    fin.close();
    fout.close();
    return 0;
}
```

> 程序中涉及到的文件操作，后续章节会做详细讲解，读者只需领悟 rdbuf() 函数的用法即可。

仍以前面创建好的 in.txt 文件为例，执行此程序后，控制台不会输出任何数据，而是会在该项目的目录下生成一个 out.txt 文件，其中就存有该程序的执行结果：

```
C++
http://c.biancheng.net/cplus/
```

C++通过控制台实现重定向

以上 2 种方法，都是从代码层面实现输入输出流的重定向。除此之外，我们还可以通过控制台实现输入输出的重定向。

举个例子，假设有如下代码（文件名为 demo.cpp）：

```
#include <iostream>
```

```
#include <string>
using namespace std;
int main()
{
    string name, url;
    cin >> name >> url;
    cout << name << '\n' << url;
    return 0;
}
```

通过编译、链接后，会生成一个 demo.exe 可执行文件，该执行文件可以双击执行，也可以在控制台上执行。例如，打开控制台（Windows 系统下指的是 CMD命令行窗口，Linux 系统下指的是 Shell 终端），并输入如下指令：

```
C:\Users\mengma>D:\demo.exe
C++ http://c.biancheng.net/cplus/
C++
http://c.biancheng.net/cplus/
```

可以看到，demo.ext 成功被执行，但程序中的 cin 和 cout 并没有被重定向，因此这里仍需要我们手动输入测试数据。

在此基础上，继续在控制台执行如下指令：

```
C:\Users\mengma>D:\demo.exe <in.txt >out.txt
```

> 需要注意的是，执行此命令前，需保证 C:\Users\mengma 目录下有 in.txt 文件。

执行后会发现，控制台没有任何输出。这是因为，我们使用了"<in.txt"对程序中的 cin 输入流做了重定向，同时还用 ">out.txt"对程序中的 cout 输出流做了重定向。

如果此时读者进入 C:\Users\mengma 目录就会发现，当前目录生成了一个 out.txt 文件，其中就存储了 demo.ext 的执行结果。

> 在控制台中使用 > 或者 < 实现重定向的方式，DOS、windows、Linux 以及 UNIX 都能自动识别。

优秀文章

MySQL删除数据库（DROP DATABASE语句）	MySQL CHAR、VARCHAR、TEXT、ENUM、SET（字符串类型）
什么是队列（队列存储结构）	DNS报文格式解析（非常详细）
Linux系统安全性分析	Linux PATH环境变量是什么，有什么用？（入门必读）
Django用户认证系统权限管理	Nexus索引与构件搜索
什么是并发（非常详细）	Spring框架介绍（非常详细）