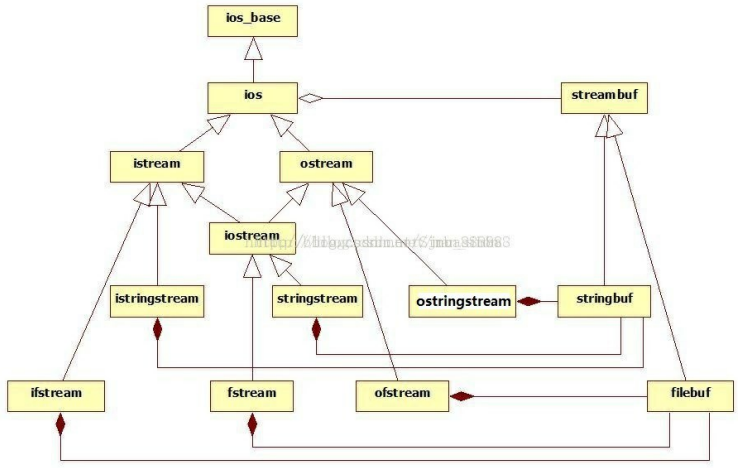
原创 mans-men 于 2017-09-27 11:14:47 发布 18994 收藏 31 版权
分类专栏： c++ 文章标签： stream 流 streambuf

在C++ 中引入了流的概念，我们很方便的通过流来读写文本数据和二进制数据，那么流对象的数据究竟是怎么存储的呢，为了搞清这个问题，先来看一看c++ 的 io 体系：



由图可以看出，在 stream 的实现中，除了虚基类IOS_BASE之外，所有的类内部都有一个streambuf, streambuf 是一个虚基类（不能被实例化，因此所内部包含streambuf(这个虚基类而非其子类)的类也是虚基类），代表流对象内部的缓冲区，就是我们流操作中输入输出的内容在内存中的缓冲区。

Streambuf有两个子类，分别是stringbuf 和 filebuf,这两个子类可以被实例化，我们常用的文件流和字符串流，内部的缓冲区就是这两个类。

我们平常使用到的流基本是标准输入输出流，文件流和字符串流。在每个流初始化的时候都会初始化相应的streambuf(其实是它的子类)用来缓冲数据。

当我们用文件或者字符串初始化流的时候，流内部会保存该文件和字符串的信息，而在内部实例化一个streambuf用来缓冲数据，些数据时，当缓冲区满的时候再将数据写到文件或者字符串，读数据时当缓冲区没有数据时从文件或字符串读数据到缓冲区。

在文件流这种情况下，streambuf 是为了避免大量的IO 操作

在字符串流的情况下，streambuf （其实是套在上面的流对象）是为了提供字符串的格式化读取和输出操作（想象字符串是你从键盘输入的数据）

所以streambuf 可以看作一块缓冲区，用来存储数据，在这种情况下，我们常常在程序中用的 char数组缓冲区是不是可以被替代呢？答案是of course

而且，有了streambuf ,缓冲区的管理和写入写出都非常方便，最好的是流对象有复制拷贝等构造函数可以方便参数传递等需要拷贝的情景。

但是streambuf 本身是个虚基类，不能实例化，所以要用streambuf 就需要自己继承streambuf 写一个新的类出来才能用，这个实现方法最后介绍，好在c++ 标准类库实现了两个子类stringbuf 和 filebuf ,所以我们可以选stringbuf 来作为我们的数据缓冲对象（不选filebuf 是因为它的实现和文件紧耦合的，只适合文件流）

流对象有一个构造函数是通过streambuf 来构造:

```
stringbuf sb;
istream is(&sb);
```

有了流对象我们就可以在流上进行各种输入输出操作，输入会从缓冲区读数据，输出会将数据写到缓冲区

注意对缓冲区的读写一定要注意方法，流符号是格式话输入输出，get,put,read,write等是二进制读写。

格式化输入的内容应当格式化读取，二进制写入应当二进制读取否则会出现写入和读出数据不一致的问题

格式化写入一个int 数据时，会将该数据每位分离出来，按照字符编码写到缓冲区，例如 int x= 123, 格式化写入以后缓冲区存以后，缓冲区有三个字节分别存放1，2，3的字符编码。格式化读出是相反的过程，将读到的字符转成相应的类型的数据

二进制写入时进行直接的内存拷贝不做任何动作，例如 int x = 123 二进制写入后（二进制写时需要取地址，转成char* 并指出要写入的字节数，如f.write((char*)&x,sizeof(int))

写完后缓冲区的数据是0x0000007b,是计算机内存中对123 的内存的完全拷贝

下面是缓冲区使用的情景：

考虑一个生产者，消费者的问题，线程A 生成的数据，线程B读取，可以解决的方案如下：

1. 设立全局变量，缓冲数据，A,B都可以访问（在这种情况下，A 生产的时候要考虑缓冲区是否够用，B读取的时候要判断当前是否有有效数据可读，而且很难设计一个合理分配内存的缓冲区（想象A生产的数据有时很大，有时很小））

2.网络通信（TCP,UDP）

3. streambuf 登场，有了streambuf配合stream， A就像正常操作流一样往流对象里塞数据，而B 就像正常操作流一样从流里面读数据，不用关心其他问题，只要这两个流的sterambuf 是同一个对象。

上一段代码：

```
#include <iostream>
#include <streambuf>
#include <sstream>
#include <fstream>
#include <string>
#include <cstring>
#include <memory>
#include <thread>
using namespace std;
stringbuf buf;
istream in(&buf);
ostream out(&buf);
bool flag = false;
void threadb() {
    char data;
```

```
    while (true) {
        if (flag) {
     in  >>  data;
      cout  <<  "thread B recv:" << data << endl;
      flag  =  false;
        }
    }
}
int main() {
    thread consumer(threadb);
    char data;
    while (true) {
   cin  >>  data;
   out  <<  data;
    flag  =  true;
    }
    return 0;
}
```

在特殊的情景下可以实现自己的streambuf类，自己实现的类必须继承streambuf 类，自定义的streambuf 必须实现overflow,underflow,uflow等方法，其中overflow在输出缓冲区不够用时调用，underflow和uflow在输入缓冲区无数据时调用,区别是uflow 会让读取位置前进一位，而underflow不会。sreambuf 内部维护着六个指针 eback,gptr,egptr,pbase,pptr,epptr.分别指向读取缓冲区的头，当前读取位置，尾，写缓冲区的头，当前写位置，尾（实际上这几个指针指向同一段缓冲区）

自定义实现方式要注意要在该返回EOF的时候，返回EOF,underflow和uflow都有可能返回EOF,一旦返回了EOF则标志着流结束，之后对流的操作无效。

如下代码实现了一个自定义的streambuf:

```
#include <fstream>
#include <string>
#include <cstring>
#include <memory>
using namespace std;
class mybuf : public streambuf {
public:
    enum{ SIZE = 10};
    mybuf() {
        memset(buffer, 'j', 10);
        //buffer[3] = ' ';
        setbuf(buffer, SIZE);
    }
    void log() {
   cout  <<hex<<gptr() << endl;
    }
protected:
    int_type overflow( int_type c) {
   cout  <<  "overflow" << endl;
        return c;
    }
    streambuf* setbuf(char* s, streamsize n) {
        setp(s, s + n);
        setg(s, s, s + n);
        return this;
    }
    int_type underflow() override{
   cout  <<  "here"<<endl;
        memset(buffer, 'w', 10);
        setg(buffer, buffer, buffer+10);
        return ' ';
    }
    int_type uflow() override{
   cout  <<  "uflow" << endl;
        memset(buffer, 'x', 10);
        setg(buffer, buffer, buffer + 10);
        return EOF;
    }
private:
    char buffer[SIZE];
};
int main() {
 mybuf buf;
    char test[2000];
    memset(test, 'a', 2000);
    //buf.pubsetbuf(test, 1000);
 string hh;
 string xx;
    istream in(&buf);
    ostream tt(&buf);
 in>>hh;
 cout << hh << endl;
    //tt.write(test, 9);
 in >> xx;
 in .read(test, 11);
 cout<< xx << endl;
 cout << "end" << endl;
    return 0;
}
```

35 原创 | 18万+ 周排名 | 84万+ 总排名 | 18万+ 访问 | 等级
1654 积分 | 29 粉丝 | 72 获赞 | 31 评论 | 202 收藏
私信 关注

搜博主文章

热门文章

最新评论

您愿意向朋友推荐“博客详情页”吗？

强烈不推荐 不推荐 一般般 推荐 强烈推荐

最新文章

2018年 1篇 2017年 37篇
2015年 1篇

C/C++编程：流缓冲类std::basic_streambuf — OceanStar的博客 1157

C++ 流（stream）总结 热门推荐 — luguifang2011的专栏 3万+
C++中流的操作总结

Boost.Asio的streambuf注意事项 — weixin_44048823的博客 217
Boost.Asio的streambuf注意事项 假设有boost::asio::streambuf buf 使用std::ostream(&buf)向buf写数据时，如果使用<<符号，那么数字会在网络传输时被…

asio::streambuf的使用 — audi2的专栏 783
这个神器设计的非常巧妙，它有两个流缓冲区。一个input序列，一个output序列。比较难理解的是，在streambuf内部看来，它需要读取output序列，写入i…

关于C++之streambuf — itzyjr的专栏 652
typedef basic_streambuf<char> streambuf; typedef basic_streambuf<wchar_t> wstreambuf; <streambuf>提供streambuf缓冲区类，与输入/输出流结合…

C++使用streambufs实现输入输出的复制、加载、重定向和tee — 大草的博客 662
streambuf 复制、加载、重定向和 tee 流的示例

输入输出流和streambuf的使用 — musezh的博客 712
C++ IO 体系 ios_base定义了同字符类型无关的属性和操作，istream和ostream分别定义了同输入和输出相关的操作，iostream同时支持输入和输出。它们…

C++中的文件file和流缓冲streambuf操作 — halazi100 5784
1.引入头文件fstream #include <fstream> fstream头文件定义了用于文件输入的类ifstream和文件输出的类ofstream 参考文档 http://www.cplusplus.co…

streambuf.h — 03-23
当你遇到如下情况时, fatal error C1083: Cannot open include file: 'streambuf': No such file or directory 就是因为少了头文件. 里面一共有三个文.exceptio…

[C++]将标准IO库应用于套接字 — weixin_30919235的博客 146
最近在写一个网络程序，需要将字符串一行一行地写入套接字，再一行一行地从套接字读取出来。由于没有现成的函数来以行的方式来操作套接字，只能…

aws php sdk s3 实例,aws sdk cpp中S3相关的应用和示例 — weixin_35032509的博客 352
概述aws提供多种sdk去访问S3，包括java、go、php、js、ruby、net、c++等，本篇文章结合作者最近应用的实践，介绍aws sdk cpp中访问S3的使用，…

开源框架PhxRPC（一）之streambuf — qq_18661257的专栏 744
提要 在讲解PhxRPC的之前，介绍streambuf是必要的，本篇会带大家走一遍 steambuf继承重写流程，分别实现两套用缓冲区的流操作读入写出。 一个是…

C++自定义缓冲区streambuf — letterwuyu的专栏 8622
Straem缓冲区其接口由class basic_treambufC++对于文件的操作是通过fstream创建文件流来实现的， 它不支持文件描述符，对于这个问题可以通过缓冲…

A beginner's guide to writing a custom stream buffer (std::streambuf) — 信箱神、得引擎 3271
原文：http://www.mr-edd.co.uk/blog/beginners_guide_streambuf Streams are one of the major abstractions provided by the STL as part of the C++ sta…

C++流扩展 streambuff-转载 — Butterfly_Dreaming的专栏 405
C++流扩展 streambuff结合socket流的扩展参考链接 结合socket流的扩展 #include <WinSock2.h> ///// out //no buffer class SocketOutStreamBuf : public …

“相关推荐”对你有帮助么？

非常没帮助 没帮助 一般 有帮助 非常有帮助