关于提升编译速度的代码编写要点

文档创建日期:2013年11月22日

文档作者：洪坤安

这次得蒙有机会对gameserver进行针对编译速度的代码整理，深感荣幸。本次整理共耗时2天半，11\_20日开始，11月22日结束（附：这之后于11月27号又针对编译速度进行了 一次调整），整理完后我觉得有必要做个小记录。是以有此文档。本项目服务端的同学可以只看第二小节。

目录

[一、这次代码整理的思路 1](#_Toc373478648)

[二、关于提升编译速度的代码编写要点 5](#_Toc373478649)

[三、预编译头文件技术 7](#_Toc373478650)

## 一、这次代码整理的思路

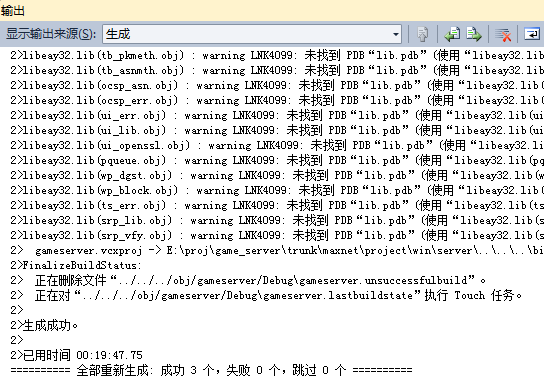
，本次针对gameserver编译速度的代码整理发现诸多问题

1. <gsapplication.h>在项目中被187个cpp文件所包含（整个gameserver项目也就396个cpp文件，将近一半文件），而经统计，大部分文件只使用到了CGSApplication的GetDBMgr接口，却被迫包含了庞大的<gsapplication.h>头文件。我在<gsapplication.h>文件中只添加一个空格，结果有188个文件需要重新编译。
2. <player.h>被23个.h头文件包含，而这23个头文件又分别被不同的头文件包含，形成一条错综复杂的包含链，我在<player.h>中只加一个空格，结果整个gameserver有359个文件需要重新编译。
3. <player.h>中的许多成员对象采取直接持有的方式，也因此，头文件中必须包含相应对象的头文件，比如，为了持有装备精通类，如果以CEquipMaster m\_oMaster的持有方式，则必须#include <equipmaster.h>。在这种情况下，<player.h>>包含了近60个其他头文件。
4. <protocligs.h>文件被126个.h头文件包含，作为一个协议文件，被这么多的头文件引用是极不合理的。除了某些必要的情况，如直接持有<protocligs.h>中的结构体，或用到了其中的#define定义，最通常的情况是对协议结构体进行修改、存入和取出，而这些只需要传引用或指针就可以了。现在的问题是，我在<protocligs.h>文件中只添加一个空格，结果有382个文件需要重新编译。
5. 有些头文件明显有拷贝粘贴的痕迹，比如#include <sdtime.h>、#include <sdstring.h>、#include <sdhashmap.h>这三个文件就经常在其他.h头文件中一起出现，达44次，而大部分都只需要用到其中的一个就够了。我大概统计下，约只有7、8个头文件需要真的包含到<sdtime.h>。

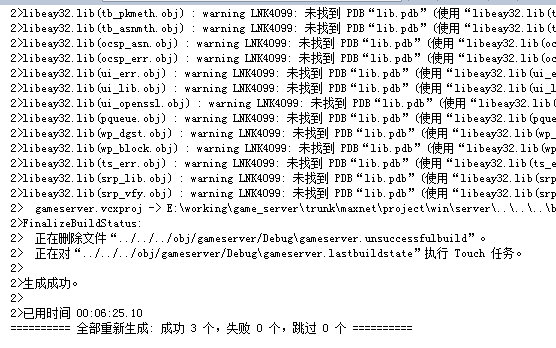
所以，我针对以上几点删除了大量的#include语句，并作了适当的类调整和接口封装，编译速度提升效果显著。可以看下表对编译速度的前后对比。（本机测试：i3、8GB内存）（附：这之后于11月27号又针对编译速度进行了 一次调整，下面另行列出）

|  |  |  |  |
| --- | --- | --- | --- |
|  | 编译耗时（不含链接） | 链接耗时 | 总耗时 |
| 整理前 | 8分58秒 | 10分49秒 | 19分47秒 |
| 整理后 | 4分40秒 | 1分45秒 | 6分25秒 |

有图有真相，下图是整理前的



下图是11月22日整理后的



具体的代码整理措施有几点

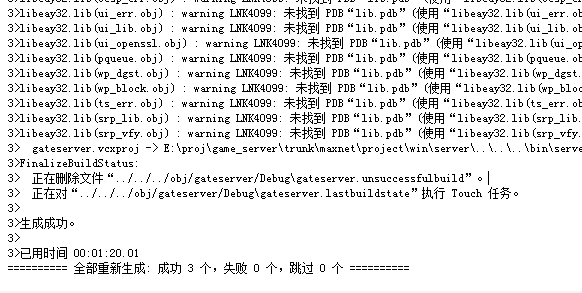
1. 鉴于大多数文件对CGSApplication所进行的只有查询和获取操作（比如GetDBMgr、GetZoneId接口），对CGSApplication类的定义毫不关心，故新建<gsapi.h>和<gsapi.cpp>，将CGSApplication的常用接口封装起来。所以，原本的((CGSApplication\*)SDGetApp())->GetDBMgr()变为gsapi::GetDBMgr()，当需要获取到DBMgr时，通过#include <gsapi.h>，而不是包含庞大的<gsapplication.h>
2. 将CPlayer类中的部分成员变量由直接对象持有改为指针持有，并将原本的自动分配成员内存改为手动通过对象池分配。
3. 头文件中尽量移除#include <player.h>，转用class CPlayer，通过前置声明降低依赖。
4. 大量头文件中包含<sdstring.h>和<sdtime.h>却根本没用到里面的任何定义，移之
5. 大部分包含<protocligs.h>协议的头文件，并未直接使用到<protocligs.h>中的结构体定义，而是以引用的方式传递，碰上这种我直接通过前置声明来移除#include <protocligs.h>
6. 尽可能将头文件包含拆分得更细，比如，<logic/base/basedefine.h>中包含了<map>、<set>、<sdhashmap.h>、<protocommondata.h>四个头文件，现有一个头文件包含了<basedefine.h>却只用到了其中的<sdhashmap.h>，碰上这种情况，直接将<basedefine.h>换成<sdhashmap.h>
7. 编译选项<是否生成清单>由/Manifest:Yes改为/Manifest:No

附：

这次调整后，编译总耗时加快到约7分钟，还算可以接受，但由于部分整理代码未合要求，所以11月23号我又将部分代码撤了回去，于是编译时间又变为了将近11分钟，成效因而寥寥。然而，11月27号，采用了预编译头技术后，项目的编译时间再次大幅提升，此次编译速度优化任务才算圆满完成。

下表是采用预编译技术前后的gameserver、gateserver、gmserver3个项目重新编译的耗时对比。（本机测试：i3、8GB内存）

|  |  |  |  |
| --- | --- | --- | --- |
|  | 编译耗时（不含链接） | 链接耗时 | 总耗时 |
| 未采用预编译头技术 | 5分10秒 | 5分24秒 | 10分34秒 |
| 采用预编译头技术后 | 1分20秒 | 10秒 | 1分30秒 |



所以，最终的编译总耗时由20分钟优化到1分钟半。（后来升级到i5之后变为了1分钟整）

## 二、关于提升编译速度的代码编写要点

关于提升编译速度，除了类和接口的设计上要注意之外，还要考虑的就是头文件的互相包含问题了。我感觉现在项目还不是很意识到这点，所以写下几点要点以供参考，编译速度影响到的是整个项目，所以完全有必要在此方面互相多做监督。

如下：

1. 尽量不要出现#include <gsapplication.h>语句，请优先考虑用#include <gsapi.h>代替

如：当你想用((CGSApplication\*)SDGetApp())->GetDBMgr()时，先看看能不能用gsapi::GetDBMgr()

1. 能用前置声明就绝不#include

以<enemy.h>为例，里面只有一个方法用到了<protocligs.h>中的结构体定义，原本的文件是：

|  |
| --- |
| #include <protocligs.h>  BOOL GetDT\_ENEMY\_DATA\_CLI(DT\_ENEMY\_DATA\_CLI&); |

由于其中的DT\_ENEMY\_DATA\_CLI只是引用，故可以改成：

|  |
| --- |
| struct tagDT\_ENEMY\_DATA\_CLI;  typedef tagDT\_ENEMY\_DATA\_CLI DT\_ENEMY\_DATA\_CLI;  BOOL GetDT\_ENEMY\_DATA\_CLI(DT\_ENEMY\_DATA\_CLI&); |

1. <player.h>尽量不要出现#include

CPlayer玩家类贯穿游戏逻辑的始终，大量其余cpp文件包含<player.h>，每加一个包含将意味着几百个cpp文件同时包含。

关于这点，请采用之前我们服务端通行的做法：如果想加一个成员变量，以指针的方式来持有。

如，现在想在CPlayer加一个CBag成员，可以把CBag m\_oBag 改为Cbag\* m\_poBag，并创建CBagMgr类通过对象池的方式来为m\_poBag分配和回收内存。详细可以参照已有代码。

总而言之，唯一准则就是：尽量使在.h头文件中的内容达到最少。以上几个要点不只是为了优化编译速度，也在于这是合理的编程准则，头文件中本来就不应暴露太多信息和额外接口，无谓的#include更是一种语法上的强行耦合，所以最需谨慎。编程中有一种叫信息隐藏的思想，即：

在设计模块时，使得该模块包含的信息，对于不需要这些信息的模块，是不可访问的。

就其寓意来说，是比面向对象的抽象程度更高的，也更为通行。也因此，才有了pimpl这类契合信息隐藏的接口设计方式。我们项目虽然不用像pimpl那么严格，不过平常编写的时候最好多注意一下。

现在项目的代码还是存在头文件内#include语句太多的现象，我这次整理只能取其中一部分比较臃肿的头文件来删改，剩下还有很多遗漏，包括之后提交的代码可能也再次重蹈覆辙，所以，希望我们服务端的同学碰到时能顺手改一改。平常提交代码时也请详加注意。

话说，我觉得现在项目的代码还算好维护，就是编译起来慢了点，这次整理代码虽然很辛苦，纯粹是体力活，但也学到了很多。所以能有这个机会也要感谢项目组。我们项目虽然有时候会暴露一下小问题，不过我们一起来改善，一起成长，最终一定会朝好的方向发展，这点是不用怀疑的。哈哈。

（对于以上3个要点如果有疑问，可以做极限假设的方式来考虑，以第一点为例，上面说推荐用#include <gsapi.h>来取代#include <gsapplication.h>，单纯这个做法可能比较难证明效果，但现在假设，<gsapplication.h>里面包含了100000000个其他头文件，效果就出来了）

附：

11月27日给项目添加预编译头支持后，编译速度已经提高了很多，以致于上述3点要点已经没有了明显效果，不过，即便是仅从代码和编程准则的角度来考虑，这些小点还是具有参考意义的，所以，还是希望各位服务端同学仍然能够按照这些原则来编程。

## 三、预编译头文件技术

由于本次采用预编译技术成效卓绝，所以特此略加介绍该技术。

（以下内容百度而来，想看更详细的可以再去查）

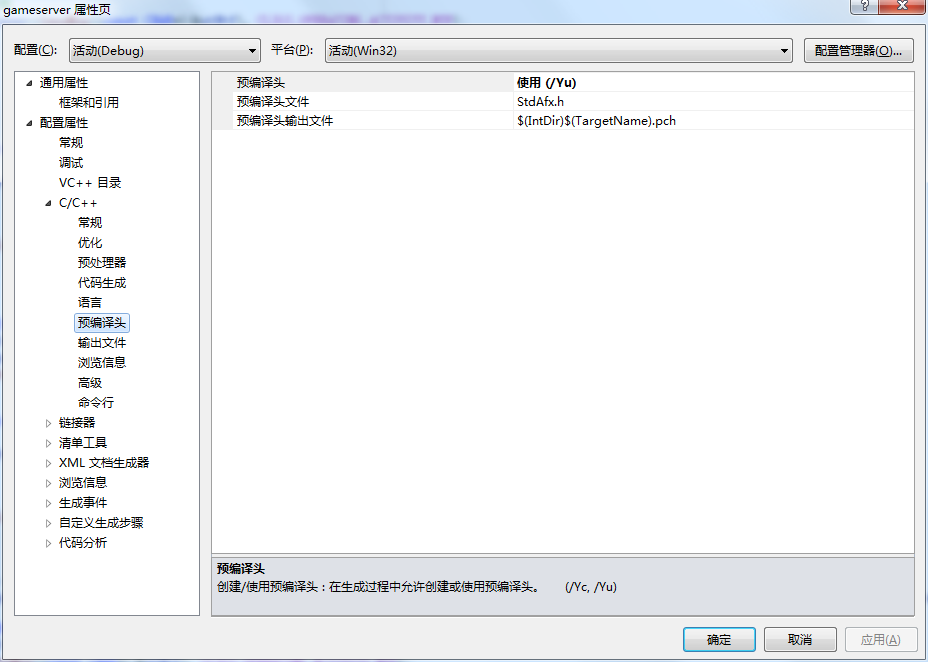
在项目中，有一些头文件是经常被其余文件包含的，由于编译器都是以文件为单位进行编译的，所以，一些经常被引用的头文件就会被反复解释和分析，于是就衍生出了预编译头文件技术。所谓预编译头，就是把头文件事先编译成一种二进制的中间格式，供后续的编译过程使用。所以，当出现大量的cpp文件均需要相同的头文件时，就可以考虑采用预编译技术了（比如像我们项目这种情况就很典型，编译速度一下提升十倍）。

关于预编译头的支持，不同编译工具的实践方式不同，但核心步骤都是先建立预编译头文件和对应cpp文件，将常用的头文件放入预编译头文件中后，让项目内所有cpp文件均包含该头文件。比如，现假设预编译头文件名为stdafx.h（这是通行的命名），则项目中所有cpp文件均需在开头加上#include <stdafx.h>。一般，如果文件少的话，可以手动添加#include <stdafx.h>，但文件多了就不好办了，不过由于我们项目是vs2010环境，所以这一过程可以适当简化，不需要修改项目cpp文件，节省了大量功夫。

下面简要介绍一下vs2010环境下添加预编译头支持的技术实践步骤。（基本跟网上的差不多，不过注意到由于利用了vs开发工具的强制包含头文件功能，为gameserver、gateserver、gmserver3个项目几百个cpp文件添加预编译支持变得非常快捷）

流程如下：

1. 首先，为项目开启预编译头支持，右键项目--属性--C/C++--预编译头选项改为使用

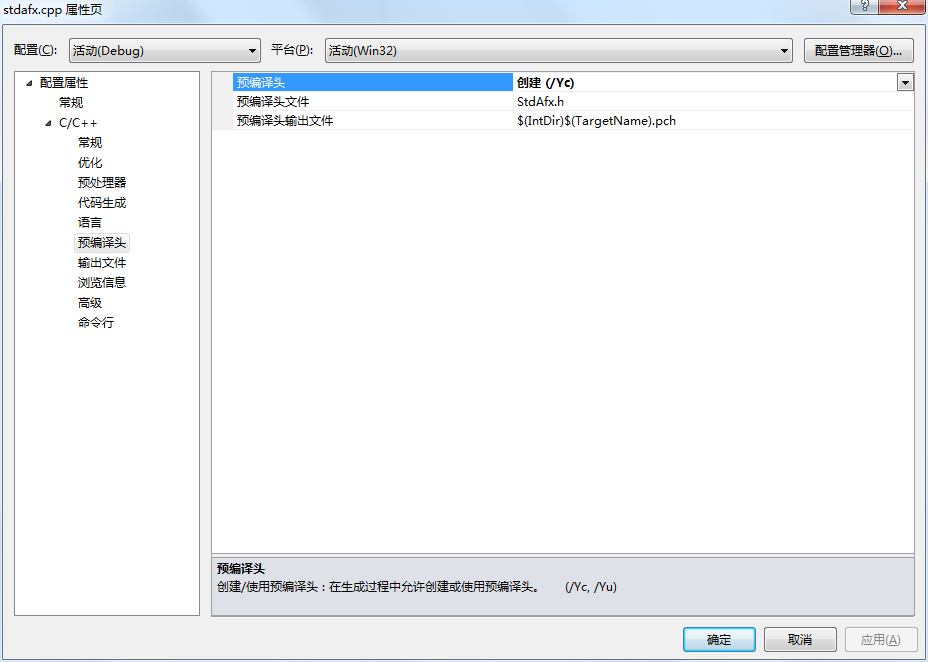


2. 创建预编译头文件stdafx.h（该命名最为通行），将最常被使用的头文件包含进去，并且，创建stdafx.cpp，里面的内容是#include <stdafx.h>

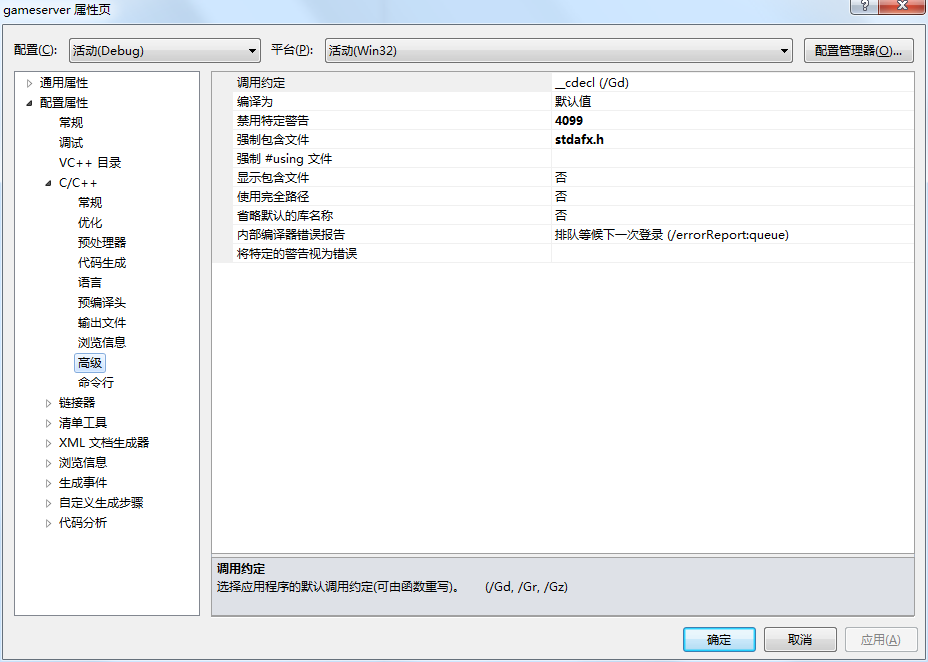
|  |
| --- |
| ///<------------------------------------------------------------------------------  //< @file: stdafx.h  //< @date: 2013年11月27日 20:06:22  //< @brief: 预编译头文件，勿删  //< @copyright:Copyright (c) 2013 heihuo. All rights reserved.  ///<------------------------------------------------------------------------------  #ifndef \_stdafx\_h\_  #define \_stdafx\_h\_  #include <logic/player/player.h>  #include <protocommondata.h>  #include <protorechargegs.h>  #include <protogtcommon.h>  #include <protocligs.h>  #include <protogmgs.h>  #include <protogsdb.h>  #include <protogsls.h>  #include <protogsns.h>  #include <protogtgs.h>  #endif // \_stdafx\_h\_ |

|  |
| --- |
| ///<------------------------------------------------------------------------------  //< @file: stdafx.cpp  //< @date: 2013年11月27日 20:08:14  //< @brief: 预包含预编译头文件，勿删  //< @copyright:Copyright (c) 2013 heihuo. All rights reserved.  ///<------------------------------------------------------------------------------  #include "stdafx.h" |

3. 在stdafx.cpp上，右键属性--C/C++--预编译头选项由使用转为创建



4. 右键项目属性，C/C++--高级选项--强制包含文件中输入stdafx.h（很重要的一步，节省了大量功夫）



自此，已为项目添加了预编译支持。网络上有条建议经常出现：不要预编译常发生修改的头文件，顾虑于此，我才迟迟不采用此技术，因为<player.h>和<protocligs.h>虽然涵盖很广，却也可以说是项目中最频繁改动的文件，因此觉得不适合。如今实践之后出现如此反差，回头再来看这句话，显见不周到之处，也有些莫名其妙。

总之，按我的说法，就是：不要预编译冷清头文件。

完