[Google gflags使用说明](http://blog.csdn.net/lezardfu/article/details/23753741)

分类： [程序库使用](http://blog.csdn.net/lezardfu/article/category/2194891)2014-04-15 13:29 4580人阅读 [评论](http://blog.csdn.net/lezardfu/article/details/23753741#comments)(0) [收藏](javascript:void(0);) [举报](http://blog.csdn.net/lezardfu/article/details/23753741#report)

[google](http://www.csdn.net/tag/google)[c++](http://www.csdn.net/tag/c%2b%2b)[开源](http://www.csdn.net/tag/%e5%bc%80%e6%ba%90)[库](http://www.csdn.net/tag/%e5%ba%93)

gflags是什么：

gflags是google的一个开源的处理命令行参数的库，使用c++开发，具备python接口，可以替代getopt。

gflags使用起来比getopt方便，但是不支持参数的简写（例如getopt支持--list缩写成-l，gflags不支持）。

如何安装使用gflags：

安装：请访问地址<https://code.google.com/p/gflags/>，下载最新版的gflags，编译安装。

使用：

1. 首先需要include "gflags.h"（废话，-\_-b）
2. #include <gflags/gflags.h>
3. 将需要的命令行参数使用gflags的宏：DEFINE\_xxxxx(变量名，默认值，help-string) 定义在文件当中，注意全局域哦。gflags支持以下类型：
   * DEFINE\_bool: boolean
   * DEFINE\_int32: 32-bit integer
   * DEFINE\_int64: 64-bit integer
   * DEFINE\_uint64: unsigned 64-bit integer
   * DEFINE\_double: double
   * DEFINE\_string: C++ string
4. 在main函数中加入：（一般是放在main函数的头几行，越早了解用户的需求越好么^\_^）
5. google::ParseCommandLineFlags(&argc, &argv, true);
6. argc和argv想必大家都很清楚了，说明以下第三个参数的作用：
7. 如果设为true，则该函数处理完成后，argv中只保留argv[0]，argc会被设置为1。
8. 如果为false，则argv和argc会被保留，但是注意函数会调整argv中的顺序。
9. 这样，在后续代码中可以使用FLAGS\_变量名访问对应的命令行参数了
10. printf("%s", FLAGS\_mystr);
11. 最后，编译成可执行文件之后，用户可以使用：executable --参数1=值1 --参数2=值2 ... 来为这些命令行参数赋值。
12. ./mycmd --var1="test" --var2=3.141592654 --var3=32767 --mybool1=true --mybool2 --nomybool3

这里值得注意的是bool类型命令行参数，除了可以使用--xxx=true/false之外，还可以使用--xxx和--noxxx后面不加等号的方式指定true和false

gflags进阶使用：

1. 在其他文件中使用定义的flags变量：有些时候需要在main之外的文件使用定义的flags变量，这时候可以使用宏定义DECLARE\_xxx(变量名)声明一下（就和c++中全局变量的使用是一样的，extern一下一样）
   * DECLARE\_bool: boolean
   * DECLARE\_int32: 32-bit integer
   * DECLARE\_int64: 64-bit integer
   * DECLARE\_uint64: unsigned 64-bit integer
   * DECLARE\_double: double
   * DECLARE\_string: C++ string
   * 在gflags的doc中，推荐在对应的.h文件中进行DECLARE\_xxx声明，需要使用的文件直接include就行了。
2. 检验输入参数是否合法：gflags库支持定制自己的输入参数检查的函数，如下：
3. static bool ValidatePort(const char\* flagname, int32 value) {
4. if (value > 0 && value < 32768) // value is ok
5. return true;
6. printf("Invalid value for --%s: %d\n", flagname, (int)value);
7. return false;
8. }
9. DEFINE\_int32(port, 0, "What port to listen on");

static const bool port\_dummy = RegisterFlagValidator(&FLAGS\_port, &ValidatePort);

1. 判断flags变量是否被用户使用：在gflags.h中，还定义了一些平常用不到的函数和结构体。这里举一个例子，判断参数port有没有被用户设定过  
       google::CommandLineFlagInfo info;  
       if(GetCommandLineFlagInfo("port" ,&info) && info.is\_default) {  
           FLAGS\_port = 27015;  
       }
2. 定制你自己的help信息与version信息：(gflags里面已经定义了-h和--version，你可以通过以下方式定制它们的内容)
   * version信息：使用google::SetVersionString设定，使用google::VersionString访问
   * help信息：使用google::SetUsageMessage设定，使用google::ProgramUsage访问
   * 注意：google::SetUsageMessage和google::SetVersionString必须在google::ParseCommandLineFlags之前执行

参考：

* <https://code.google.com/p/gflags/>
* <https://gflags.googlecode.com/git-history/master/doc/gflags.html#flagfiles>