

当前位置: 首页 > 程序阶梯 > C/C++ > 用Google的gflags优雅的解析命令行参数

## 用Google的gflags优雅的解析命令行参数

2014年10月26日 | 标签: C/C++ Google gflags Linux | 浏览: 5,835 | 评论关闭

写了这么多年的Linux下C/C++代码，一直使用getopt\_long来解析命令行参数，同时定义一个全局的struct来保存各个命令行参数的值。虽然用得比较“繁琐”，但也安于现状。最近突然发现了Google早在多年前就开源了一个解析命令行参数的“神器”gflags。赶紧来爽一把。

### 安装

- 1、去官网下载一个最新的版本(gflags-2.1.1.tar.gz)。

<https://github.com/schuhschuh/gflags/archive/v2.1.1.tar.gz>

- 2、现在流行cmake的构建方式，gflags的最新版本也改为使用cmake了。还好我最近也刚刚学习了cmake，算是跟上了潮流。有兴趣的话，可以看《让cmake显示gcc/g++的编译信息》

```
1 [amcool@leedox soft]$ tar xzvf gflags-2.1.1.tar.gz
2 [amcool@leedox soft]$ cd gflags-2.1.1
3 [amcool@leedox gflags-2.1.1]$ mkdir build
4 [amcool@leedox gflags-2.1.1]$ cd build/
5 [amcool@leedox build]$ cmake .. -DCMAKE_INSTALL_PREFIX=/home/amcool/lo
6 [amcool@leedox build]$ make
7 [amcool@leedox build]$ make install
```

就是这么简单，安装成功了。值得注意的是，我这里新建了一个build文件夹，即采用“外部构建”的方式。这样编译过程中产生的中间文件（比如.o文件）就都放在build里，不会“污染”gflags源码，做到干干净净。

### 爽一把

- 1、既然安装好了，那赶紧来写个简单的代码来爽一把。话不多说，代码才是王道！

```
1 // demo.cpp
2 #include <iostream>
3 #include <gflags/gflags.h>
4
5 using namespace std;
6
7 DEFINE_string(confPath, "../conf/setup.ini", "program configure file");
8 DEFINE_int32(port, 9090, "program listen port");
9 DEFINE_bool(daemon, true, "run daemon mode");
10
11 int main(int argc, char** argv)
12 {
13     gflags::ParseCommandLineFlags(&argc, &argv, true);
14
15     cout << "confPath = " << FLAGS_confPath << endl;
16     cout << "port = " << FLAGS_port << endl;
17
18     if (FLAGS_daemon) {
19         cout << "run background ..." << endl;
20     }
21     else {
22         cout << "run foreground ..." << endl;
23     }
24
25     cout << "good luck and good bye!" << endl;
26 }
```

#### 欢迎订阅本站



#### 热门文章

#### 随机推荐

- C++将数组元素打散的一种方法
- 阿里云(CentOS 6.3)安装MySQL + PHP + ...
- 浅析递归函数与函数调用栈
- C/C++使用ZeroMQ的SUB/PUB模式搭建一...
- cmake自动生成protobuf代码
- ssdb c++客户端api的编译问题
- 使用cmake管理组内基础项目
- Linux gcc编译参数 —— Werror的...
- 用Google的gflags优雅的解析命令行参数
- redis集群之小试牛刀

#### 读者墙

#### 标签云

## C/C++ CentOS 6.3 cmake

Google gflags hello world hiredis hostname

Linux Linux登陆欢迎语 man man-pages min()

MySQL MySQL开机自动启动 Nginx Nginx  
开机自动启动 No manual entry pcre PHP php-  
fpm开机自动启动 pop() protobuf push()

redis SecureCRT socket ssh WordPress

wordpress主题 yum ZeroMQ 二叉排序树

二叉树 微信面试 技术宅 描述符 数据结构

旋转数组二分查找 时间复杂度 栈 程序员

面试 算法 腾讯面试 递归 阿里云

```

27 gflags::ShutDownCommandLineFlags();
28 return 0;
29 }

```

2、很明显，接下来就是要编译了。这里直接用g++写一行命令就可以编译了。但是既然学了cmake，那就“大材小用”一次吧。

```

1 project(demo)
2 cmake_minimum_required(VERSION 2.8)
3 set(CMAKE_VERBOSE_MAKEFILE on)
4
5 include_directories("/home/amcool/local/gflags-2.1.1/include")
6 link_directories("/home/amcool/local/gflags-2.1.1/lib")
7
8 add_executable(demo demo.cpp)
9 target_link_libraries(demo gflags pthread)

```

3、那当然就是编译了

```

1 [amcool@leoox demo]$ ls
2 CMakeLists.txt  demo.cpp
3 [amcool@leoox demo]$ mkdir build
4 [amcool@leoox demo]$ cd build
5 [amcool@leoox build]$ cmake ..
6 [amcool@leoox build]$ ls
7 CMakeCache.txt  CMakeFiles  cmake_install.cmake  Makefile
8 [amcool@leoox build]$ make
9 [amcool@leoox build]$ ls
10 CMakeCache.txt  CMakeFiles  cmake_install.cmake  demo  Makefile
11 [amcool@leoox build]$

```

## 设定命令行参数

1、直接运行，得到的就是我们设定的**默认参数**。（聪明的你，结合代码一看，就知道参数的默认值是什么了）

```

1 [amcool@leoox build]$ ./demo
2 confPath = ../conf/setup.ini
3 port = 9090
4 run background ...
5 good luck and good bye!

```

2、设定参数值

i) 可以用 **-参数名=参数值** 或者 **参数名=参数值** 的方式来设定参数值。

ii) 对于bool类型的参数，除了上述方式外，还可以用 **-参数名** 的方式设定为true（即不带值），使用 **-no参数名** 的方式设定为false。为了统一，我建议都使用上面的第i)种方法来设定参数。

```

1 [amcool@leoox build]$ ./demo --port=8888 --confPath=./setup.ini --da
2 confPath = ./setup.ini
3 port = 8888
4 run background ...
5 good luck and good bye!
6 [amcool@leoox build]$ ./demo -port=8888 -confPath=./setup.ini -daemo
7 confPath = ./setup.ini
8 port = 8888
9 run foreground ...
10 good luck and good bye!
11 [amcool@leoox build]$ ./demo -port=8888 -confPath=./setup.ini -daemo
12 confPath = ./setup.ini
13 port = 8888
14 run background ...
15 good luck and good bye!
16 [amcool@leoox build]$ ./demo -port=8888 -confPath=./setup.ini -nodae
17 confPath = ./setup.ini
18 port = 8888
19 run foreground ...
20 good luck and good bye!
21 [amcool@leoox build]$

```

3、从文件读入“命令行”参数

如果我们的程序比较牛逼，配置项非常多，也就是说命令行参数很多，那你每次启动都要一个一个的输入，那岂不是很麻烦？gflags已经帮我们解决了，用 `-flagfile=命令行文件` 的方式就可以了。你接着往下看，就明白了。

param.cmd就是上面说的**命令行文件**。

```
1 [amcool@leoox build]$ vi param.cmd
2 --port=8888
3 --confPath=./setup.ini
4 --daemon=true
5 [amcool@leoox build]$ ./demo --flagfile=param.cmd
6 confPath = ./setup.ini
7 port = 8888
8 run background ...
9 good luck and good bye!
10 [amcool@leoox build]$
```

怎么样，这样就不怕参数配置错误了吧。保存到文件，每次启动，就很轻松了。

#### 4、从环境变量读入参数值

gflags另外还给我们提供了 `-fromenv` 和 `-tryfromenv` 参数，通过这两个参数，我们的程序可以从环境变量中获取到具体的值。两者有什么不一样呢。你看到他们的区别仅仅是有无“try”，聪明的你一定猜到了。

- `-fromenv` 从环境变量读取参数值 `-fromenv=port,confPath` 表明要从环境变量读取 `port,confPath` 两个参数的值。但是当无法从环境变量中获取到的时候，会报错，同时程序**退出**。【注意：gflags的变量名是 `FLAGS_` 我们定义的参数名，开篇的代码里，估计细心的你已经发现了】
- `-tryfromenv` 与 `-fromenv` 类似，当参数的没有在环境变量定义时，**不退出**。

也来一个例子，一看便明了。

```
1 [amcool@leoox build]$ ./demo --fromenv=port,confPath
2 ERROR: FLAGS_confPath not found in environment
3 ERROR: FLAGS_port not found in environment
4 [amcool@leoox build]$ ./demo --tryfromenv=port,confPath
5 confPath = ../conf/setup.ini
6 port = 9090
7 run background ...
8 good luck and good bye!
9 [amcool@leoox build]$ export FLAGS_confPath=./loveyou.ini
10 [amcool@leoox build]$ export FLAGS_port=36888
11 [amcool@leoox build]$ env | grep FLAGS
12 FLAGS_port=36888
13 FLAGS_confPath=./loveyou.ini
14 [amcool@leoox build]$
15 [amcool@leoox build]$ ./demo --fromenv=port,confPath
16 confPath = ./loveyou.ini
17 port = 36888
18 run background ...
19 good luck and good bye!
20 [amcool@leoox build]$
```

## 版本号和帮助信息

我们一般使用程序的时候，都离不开两个参数 `-version` 和 `-help`。来看看上面实现的demo能否支持呢？

```
1 [amcool@leoox build]$ ./demo --version
2 demo
3 [amcool@leoox build]$ ./demo --help
4 demo: Warning: SetUsageMessage() never called
5
6 Flags from /home/thrift/program/gflags/demo/demo.cpp:
7 -confPath (program configure file.) type: string
8 default: "../conf/setup.ini"
9 -daemon (run daemon mode) type: bool default: true
10 -port (program listen port) type: int32 default: 9090
```

哈，help支持了，但是version没支持，而且help信息里面还有waring。没关系，我们可以用 `SetVersionString()` 和 `SetUsageMessage()` 方法来满足需求。修改后的代码如下：

【注意：`SetVersionString()` 和 `SetUsageMessage()` 一定要在 `ParseCommandLineFlags()` 之前设定。】

```
1 #include <iostream>
2 #include <gflags/gflags.h>
```

```

3
4 using namespace std;
5
6 DEFINE_string(confPath, "../conf/setup.ini", "program configure file");
7 DEFINE_int32(port, 9090, "program listen port");
8 DEFINE_bool(daemon, true, "run daemon mode");
9
10 int main(int argc, char** argv)
11 {
12     gflags::SetVersionString("1.0.0.0");
13     gflags::SetUsageMessage("Usage : ./demo ");
14     gflags::ParseCommandLineFlags(&argc, &argv, true);
15
16     cout << "confPath = " << FLAGS_confPath << endl;
17     cout << "port = " << FLAGS_port << endl;
18
19     if (FLAGS_daemon) {
20         cout << "run background ..." << endl;
21     }
22     else {
23         cout << "run foreground ..." << endl;
24     }
25
26     cout << "good luck and good bye!" << endl;
27
28     gflags::ShutDownCommandLineFlags();
29     return 0;
30 }

```

可以来炫一把了：

```

1 [amcool@leoox build]$ ./demo --version
2 demo version 1.0.0.0
3 [amcool@leoox build]$ ./demo --help
4 demo: Usage : ./demo
5
6 Flags from /home/amcool/program/gflags/demo/demo.cpp:
7   -confPath (program configure file.) type: string
8     default: "../conf/setup.ini"
9   -daemon (run daemon mode) type: bool default: true
10  -port (program listen port) type: int32 default: 9090
11
12
13
14 Flags from /home/amcool/soft/gflags-2.1.1/src/gflags.cc:
15   -flagfile (load flags from file) type: string default: ""
16   -fromenv (set flags from the environment [use 'export FLAGS_flag
17     type: string default: ""
18   -tryfromenv (set flags from the environment if present) type: st
19     default: ""
20   -undefok (comma-separated list of flag names that it is okay to
21     the command line even if the program does not define a flag wi
22     name. IMPORTANT: flags in this list that have arguments MUST
23     flag=value format) type: string default: ""
24
25 Flags from /home/amcool/soft/gflags-2.1.1/src/gflags_completions.c
26   -tab_completion_columns (Number of columns to use in output for
27     completion) type: int32 default: 80
28   -tab_completion_word (If non-empty, HandleCommandLineCompletions
29     hijack the process and attempt to do bash-style command line f
30     completion on this value.) type: string default: ""
31
32 Flags from /home/amcool/soft/gflags-2.1.1/src/gflags_reporting.cc:
33   -help (show help on all flags [tip: all flags can have two dashe
34     type: bool default: false currently: true
35   -helpfull (show help on all flags -- same as -help) type: bool
36     default: false
37   -helpmatch (show help on modules whose name contains the specifi
38     type: string default: ""
39   -helpo (show help on the modules named by this flag value) type
40     default: ""
41   -helppackage (show help on all modules in the main package) type
42     default: false
43   -helpshort (show help on only the main module for this program)
44     default: false
45   -helpxml (produce an xml version of help) type: bool default: fa
46   -version (show version and build info and exit) type: bool defau

```

## 简单讲解如何使用gflags进行编码

有了上面的演示和代码展示，想必大家对gflags有了比较直观的认识。做了这么久的前戏，接下来，终于可以深入了解啦。请看下文《[用Google的gflags轻松的编码解析命令行参数](#)》。



« [巧用Linux下date命令互转时间戳和指定日期](#)

[用Google的gflags轻松的编码解析命令行参数](#) »

作者: leoox

除非注明，本文原创: [狮子牛](#)

如需转载，请联系本站。转载请以链接形式注明本文地址，否则进行追究！

原文链接: <http://www.leoox.com/?p=270>

相关文章

近期热评

最新日志

- Linux gcc编译参数 ---- Werror的威力
- Linux下非ROOT权限安装MySQL
- 巧用Linux下date命令互转时间戳和指定日期
- 用Google的gflags轻松的编码解析命令行参数
- 小记Linux下安装搭建RSYNC服务器
- expect + scp 免输入密码完成机器间的文件拷贝

[留言](#) [联系我](#)

Copyright © 2014-2016 [狮子牛](#) All rights reserved. [登录](#)  
Powered by [WordPress](#) & Theme by [技术宅](#). 64 Q in 0.541. [粤ICP备14039403号](#).

