[getopt与getopt\_long](http://blog.csdn.net/slmmlk2011_2/article/details/7964218)

一、getopt

getopt被用来解析命令行选项参数。

#include <unistd.h>  
       extern char \*optarg;  //选项的参数指针  
       extern int optind,   //下一次调用getopt时，从optind存储的位置处重新开始检查选项。   
       extern int opterr,  //当opterr=0时，getopt不向stderr输出错误信息。  
       extern int optopt;  //当命令行选项字符不包括在optstring中或者最后一个选项缺少必要的参数时，该选项存储在optopt中，getopt返回'？’  
       int getopt(int argc, char \* const argv[], const char \*optstring);  
 调用一次，返回一个选项。 在命令行选项参数再也检查不到optstring中包含的选项时，返回－1，同时optind储存第一个不包含选项的命令行参数。

首先说一下什么是选项，什么是参数。

字符串optstring可以下列元素，  
1.单个字符，表示选项，没有参数，optarg=NULL.  
2.单个字符后接一个冒号：表示该选项后必须跟一个参数。参数紧跟在选项后或者以空格隔开。该参数的指针赋给optarg。  
3 单个字符后跟两个冒号，表示该选项后必须跟一个参数。参数必须紧跟在选项后不能以空格隔开。该参数的指针赋给optarg。（这个特性是GNU的扩张）。

getopt处理以'-’开头的命令行参数，如optstring="ab:c::d::",命令行为getopt.exe -a -b host -ckeke -d haha   
在这个命令行参数中，-a和-h就是选项元素，去掉'-'，a,b,c就是选项。host是b的参数，keke是c的参数。但haha并不是d的参数，因为它们中间有空格隔开。

还要注意的是默认情况下getopt会重新排列命令行参数的顺序，所以到最后所有不包含选项的命令行参数都排到最后。

如果optstring中的字符串以'+'加号开头或者环境变量POSIXLY\_CORRE被设置。那么一遇到不包含选项的命令行参数，getopt就会停止，返回-1。

unistd里有个 optind　变量，每次getopt后，这个索引指向argv里当前分析的字符串的下一个索引，因此argv[optind]就能得到下个字符串，通过判断是否以 '-'开头就可。getopt返回-1后，optind指向第一个不包含选项的命令行参数（重排序后的）。

二、getopt\_long

20 世纪 90 年代，UNIX 应用程序开始支持长选项，即一对短横线、一个描述性选项名称，还可以包含一个使用等号连接到选项的参数。

GNU提供了getopt-long()和getopt-long-only()函数支持长选项的命令行解析，其中，后者的长选项字串是以一个短横线开始的，而非一对短横线。

getopt\_long() 是同时支持长选项和短选项的 getopt() 版本。下面是它们的声明：

＃i nclude <getopt.h>  
  
int getopt\_long(int argc, char \* const argv[], const char \*optstring, const struct option \*longopts, int \*longindex);  
  
int getopt\_long\_only(int argc, char \* const argv[],const char \*optstring,const struct option \*longopts, int \*longindex);

getopt\_long ()的前三个参数与上面的getopt()相同，第4个参数是指向option结构的数组，option结构被称为“长选项表”。longindex参数如果没有设置为NULL，那么它就指向一个变量，这个变量会被赋值为寻找到的长选项在longopts中的索引值，这可以用于错误诊断。

option结构在getopt.h中的声明如下：

struct option{  
     const char \*name;  
     int has\_arg;  
     int \*flag;  
     int val;  
};

对结构中的各元素解释如下：

const char \*name这是选项名，前面没有短横线。譬如"help"、"verbose"之类。int has\_arg描述了选项是否有选项参数。如果有，是哪种类型的参数，此时，它的值一定是下表中的一个。

符号常量 数值 含义

no\_argument 0 选项没有参数

required\_argument 1 选项需要参数

optional\_argument 2 选项参数可选

int \*flag如果这个指针为NULL，那么getopt\_long()返回该结构val字段中的数值。如果该指针不为NULL，getopt\_long()会使得它所指向的变量中填入val字段中的数值，并且getopt\_long()返回0。如果flag不是NULL，但未发现长选项，那么它所指向的变量的数值不变。

int val这个值是发现了长选项时的返回值，或者flag不是NULL时载入\*flag中的值。典型情况下，若flag不是NULL，那么val是个真／假值，譬如1 或0；另一方面，如果flag是NULL，那么val通常是字符常量，若长选项与短选项一致，那么该字符常量应该与optstring中出现的这个选项的参数相同。

每个长选项在长选项表中都有一个单独条目，该条目里需要填入正确的数值。数组中最后的元素的值应该全是0。数组不需要排序，getopt\_long()会进行线性搜索。但是，根据长名字来排序会使程序员读起来更容易。

以上所说的flag和val的用法看上去有点混乱，但它们很有实用价值，因此有必要搞透彻了。

大部分时候，程序员会根据getopt\_long()发现的选项，在选项处理过程中要设置一些标记变量，譬如在使用getopt()时，经常做出如下的程序格式：

int do\_name, do\_gf\_name, do\_love; /\*标记变量\*/  
char \*b\_opt\_arg;  
  
while((c = getopt(argc, argv, ":ngl:")) != -1)  
{  
     switch (c){  
     case 'n':  
         do\_name = 1;  
     case 'g':  
         do\_gf\_name = 1;  
         break;  
         break;  
     case 'l':  
         b\_opt\_arg = optarg;  
     ……  
     }  
}

当flag 不为NULL时，getopt\_long\*()会为你设置标记变量。也就是说上面的代码中，关于选项'n'、'l'的处理，只是设置一些标记，如果 flag不为NULL,时，getopt\_long()可以自动为各选项所对应的标记变量设置标记，这样就能够将上面的switch语句中的两种种情况减少到了一种。下面给出一个长选项表以及相应处理代码的例子。

清单5：  
＃i nclude <stdio.h>  
＃i nclude <getopt.h>  
  
int do\_name, do\_gf\_name;  
char \*l\_opt\_arg;  
  
struct option longopts[] = {  
     { "name",         no\_argument,             &do\_name,         1     },  
     { "gf\_name",     no\_argument,             &do\_gf\_name,     1     },  
     { "love",         required\_argument,     NULL,                 'l'     },  
     {      0,     0,     0,     0},  
};  
  
int main(int argc, char \*argv[])  
{  
     int c;  
      
     while((c = getopt\_long(argc, argv, ":l:", longopts, NULL)) != -1){  
         switch (c){  
         case 'l':  
             l\_opt\_arg = optarg;  
             printf("Our love is %s!\n", l\_opt\_arg);  
             break;  
         case 0:  
             printf("getopt\_long()设置变量 : do\_name = %d\n", do\_name);  
             printf("getopt\_long()设置变量 : do\_gf\_name = %d\n", do\_gf\_name);  
             break;  
         }  
     }  
     return 0;  
}

在进行测试之前，再来回顾一下有关option结构中的指针flag的说明吧。

如果这个指针为NULL，那么getopt\_long()返回该结构val字段中的数值。如果该指针不为NULL，getopt\_long()会使得它所指向的变量中填入val字段中的数值，并且getopt\_long()返回0。如果flag不是NULL，但未发现长选项，那么它所指向的变量的数值不变。

下面测试一下：

$ ./long\_opt\_demo --name  
getopt\_long()设置变量 : do\_name = 1  
getopt\_long()设置变量 : do\_gf\_name = 0  
  
$ ./long\_opt\_demo --gf\_name  
getopt\_long()设置变量 : do\_name = 0  
getopt\_long()设置变量 : do\_gf\_name = 1  
  
$ ./long\_opt\_demo --love forever  
Our love is forever!  
  
$ ./long\_opt\_demo -l forever  
Our love is forever!

测试过后，应该有所感触了。关于flag和val的讨论到此为止。下面总结一下get\_long()的各种返回值的含义：

返回值 含义

0       getopt\_long()设置一个标志，它的值与option结构中的val字段的值一样

1        每碰到一个命令行参数，optarg都会记录它

'?'       无效选项

 ':'        缺少选项参数 'x' 选项字符'x'

 -1       选项解析结束

从实用的角度来说，我们更期望每个长选项都对应一个短选项，这种情况下，在option结构中，只要将flag设置为NULL，并将val设置为长选项所对应的短选项字符即可。譬如上面清单5中的程序，修改如下。

清单6： ＃i nclude <stdio.h>  
＃i nclude <getopt.h>  
  
int do\_name, do\_gf\_name;  
char \*l\_opt\_arg;  
  
struct option longopts[] = {  
     { "name",         no\_argument,             NULL,                 'n'     },  
     { "gf\_name",     no\_argument,             NULL,                 'g'     },  
     { "love",         required\_argument,     NULL,                 'l'     },  
     {      0,     0,     0,     0},  
};  
  
int main(int argc, char \*argv[])  
{  
     int c;  
      
     while((c = getopt\_long(argc, argv, ":l:", longopts, NULL)) != -1){  
         switch (c){  
         case 'n':  
             printf("My name is LYR.\n");  
             break;  
         case 'g':  
             printf("Her name is BX.\n");  
             break;  
         case 'l':  
             l\_opt\_arg = optarg;  
             printf("Our love is %s!\n", l\_opt\_arg);  
             break;  
         }  
     }  
     return 0;  
}

测试结果如下：

$ ./long\_opt\_demo --name --gf\_name --love forever  
My name is LYR.  
Her name is BX.  
Our love is forever!  
  
$ ./long\_opt\_demo -ng -l forever  
My name is LYR.  
Her name is BX.  
Our love is forever! 9、在LINUX之外的系统平台上使用GNU getopt()或getopt\_long()只要从GNU程序或GNU C Library(GLIBC)的CVS档案文件中copy源文件即可（http://sourceware.org/glibc/）。所需源文件是 getopt.h、getopt.c和getoptl.c，将这些文件包含在你的项目中。另外，你的项目中最好也将COPYING.LIB文件包含进去，因为GNU LGPL（GNU 程序库公共许可证）的内容全部包括在命名为COPYING.LIB 的文件中。

**三、结论**

程序需要能够快速处理各个选项和参数，且要求不会浪费开发人员的太多时间。在这一点上，无论是GUI(图形用户交互）程序还是CUI（命令行交互）程序，都是其首要任务，其区别仅在于实现方式的不同。GUI通过菜单、对话框之类的图形控件来完成交互，而CUI使用了纯文本的交互方式。在程序开发中，许多测试程序用CUI来完成是首选方案。

getopt() 函数是一个标准库调用，可允许您使用直接的 while/switch 语句方便地逐个处理命令行参数和检测选项（带或不带附加的参数）。与其类似的 getopt\_long() 允许在几乎不进行额外工作的情况下处理更具描述性的长选项，这非常受开发人员的欢迎。