转 getopt与getopt_long

2013年10月11日 16:20:44　阅读数：2346

很多命令行程序中都有这两个文件：getopt.c与getopt_long.c。它们的作用是可以方便的获取命令行的参数。在此转一篇文章，讲述它们的作用。

一、getopt

getopt被用来解析命令行选项参数。

```cpp
#include <unistd.h>
        extern char *optarg;  //选项的参数指针，存放选项对应的输入参数
        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）。

如果optstring参数的第一个字符是冒号，那么getopt()函数就会保持沉默，并根据错误情况返回不同字符，如下：
“无效选项” —— getopt()返回'?'，并且optopt包含了无效选项字符（这是正常的行为）。
“缺少选项参数” —— getopt()返回':'。
如果optstring的第一个字符不是冒号，那么缺少选项参数时getopt()返回 '?'，这会使得这种情况不能与无效选项的情况区分开。
例如optstring为:a:b::c，表示a带一个参数,b可选,c不带参数
如果输入d，“无效选项“，getopt返回'?'
如果输入的a忘记带参数，“缺少选项参数”，getopt应返':' ;如果optstring的第一个字符不是':'的话，那么将会把这个错当成"无效参数"，从而getopt返回'?'；从而无法区别错误类型。

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。

getopt()返回我们指定的参数选项.同时将参数值保存在optarg中,变量 optind 包含下一个 argv 参数作为对 getopt() 下一次调用的索引，变量 optopt 保存最后一个由 getopt() 返回的已知的选项。如果已经分析完成所有的参数函数返回-1.这个时候optind指出非可选参数的开始位置.

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。它可以根据输入的option是单横线还是双横线开头来区分是短选项还是长选项。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：

```cpp
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。
长选项与长选项的参数之间通过空格或一个'='来隔开。

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：

```cpp
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：

```cpp
#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：

```cpp
#include <stdio.h>
#include <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
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。
更具体的用法，可以通过man查看getopt_long的帮助了。

文章标签： getopt getopt_long 命令行 选项
个人分类： 纯编程