C/C++中命令行参数的原理总结

分类： C 2009-03-10 16:21 5838人阅读 评论(0) 收藏 举报

struct c cmd null delete string

在c/c++中，命令行参数的传递是利用main进行形参传递实现

【1】了实现命令行参数我们将使用main(int argc,char* argv[])这样的形式进行定义argc和argv可以换成你自己喜欢的名称不一定要用argv，argc这些形式只是习惯而已，char* argv[]我们前面已经讲述过，这就是一个指向指针数组，argv就是一个指针数组名，argv不是常量指针，而是具备变量特性的变量指针，它是可以移动的，由 此我们可以改写成char* *argv也是正确的，int argc这个定义返回的将是参数的个数所以标记为整形(int)。

例如：编写一命令文件，把键入的字符串倒序打印出来。设文件名为invert.c

```cpp
#include<stdio.h>
#include<conio.h>
int main(int argc,char *argv[])
{
    int i;
    for(i=argc-1;i>0;i--)
        printf("%s ",argv[i]);
    getch();
        return 0;
}
```

获得你译生成后的EXE文件路径

运行－>输入CMD

先换到盘：输入盘符后加冒号

cd 路径

例如：编译后的程序在d:/work/invert.exe

运行－>cmd

d:

cd /work

invert I love china

【2】在实际程序之中我们经常要对命令行参数进行分析. 比如我们有一个程序a可以接受许多参数.一个可能的情况是

a -d print --option1 hello --option2 world

那么我们如何对这个命令的参数进行分析了?经常用函数是getopt和getopt_long。

```cpp
#include <unistd.h>
#include <getopt.h>

int getopt(int argc,char const **argv, const char *optstring);
int getopt_long(int argc,char const **argc,const char string,const strUCt option *longopts, int *lo

extern char *optarg;
extern int optind,opterr,optopt;

struct option {
    char *name;
    int has_flag;
    int *flag;
    int value;
};
```

getopt_long是getopt的扩展.getopt接受的命令行参数只可以是以(-)开头,而getopt_long还可以接受(--)开头的参数.一般以(-)开头的参数的标志只有一个字母,而以(--)开头的参数可以是一个字符串.如上面的 -d,--option1选项.

argc,和argv参数是main函数的参数.optstring指出了我们可以接受的参数.其一般的形式为:参数1[:]参数2[:].... 其中参数是我们可以接受的参数,假如后面的冒号没有省略,那么表示这个参数出现时后面必需要带参数值. 比如一个optstring为abc:d:表示这个参数选项可以为a,b,c,d其中c,d出现时候必须要有参数值.假如我们输入了一个我们没有提供的参数选项.系统将会说 不熟悉的选项. getopt返回我们指定的参数选项.同时将参数值保存在optarg中,假如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.

```cpp
#include <stdio.h>
#include <unistd.h>

int main(int argc,char **argv)
{
   int is_a,is_b,is_c,is_d,i;
   char *a_value,*b_value,*c_value,temp;
   is_a=is_b=is_c=is_d=0;
    a_value=b_value=c_value=NULL;
    if(argc==1)
    {
        fprintf(stderr,"Usage:%s [-a value] [-b value] [-c value] [-d]  arglist ... ",
        argv[0]);
        exit(1);
    }
    while((temp=getopt(argc,argv,"a:b:c:d"))!=-1)
    {
        switch (temp)
        {
            case 'a':
```

```
                is_a=1;
                a_value=optarg;
                break;
            case 'b':
                is_b=1;
                b_value=optarg;
                break;
            case 'c':
                is_c=1;
                c_value=optarg;
                break;
            case 'd':
                is_d=1;
                break;
        }
    }

    printf("Option has a:%s with value:%s ",is_a?"YES":"NO",a_value);
    printf("Option has b:%s with value:%s ",is_b?"YES":"NO",b_value);
    printf("Option has c:%s with value:%s ",is_c?"YES":"NO",c_value);
    printf("OPtion has d:%s ",is_d?"YES":"NO");
    i=optind;
    while(argv[i]) printf(" with arg:%s ",argv[i++]);
    exit(0);
}
```

getopt_long比getopt复杂一点,不过用途要比getopt广泛.struct option 指出我们可以接受的附加参数选项.

name:指出长选项的名称(如我们的option1)

has_flag:为0时表示没有参数值,当为1的时候表明这个参数选项要接受一个参数值.为2时表示参数值可以有也可以没有.

指出函数的返回值.假如为NULL,那么返回val,否则返回0.并将longindex赋值为选项所在数组(longopts)的位置.

[cpp]

```cpp
/* 这个实例是从 GNU Libc 手册上看到的 */
    #include <stdio.h>
    #include <stdlib.h>
    #include <getopt.h>


    int main (int argc, char **argv)
    {
        int c;

        while (1)
        {
            struct option long_options[] =
            {
                {"add", 1, 0, 0},
```

```
16.                 {"append", 0, 0, 0},
17.                 {"delete", 1, 0, 0},
18.                 /* 返回字符c,等同于 -c 选项 */
19.                 {"create", 0, 0, 'c'},
20.                 {"file", 1, 0, 0},
21.                 /* 数组结束 */
22.                 {0, 0, 0, 0}
23.             };
24.             /* getopt_long stores the option index here. */
25.             int option_index = 0;
26.
27.             c = getopt_long (argc, argv, "abc:d:",
28.             long_options, &option_index);
29.
30.             /* Detect the end of the options. */
31.             if (c == -1)
32.               break;
33.
34.             switch (c)
35.             {
36.                 case 0:
37.                     printf ("option %s", long_options[option_index].name);
38.                     if (optarg)
39.                     printf (" with arg %s ", optarg);
40.                     break;
41.
42.                 case 'a':
43.                     puts ("option -a ");
44.                     break;
45.
46.                 case 'b':
47.                     puts ("option -b ");
48.                     break;
49.
50.                 /* 可能是-c --creat参数指出来的 */
51.                 case 'c':
52.                     printf ("option -c with value `%s' ", optarg);
53.                     break;
54.
55.                 case 'd':
56.                     printf ("option -d with value `%s' ", optarg);
57.                     break;
58.             }
59.         }
60.
61.         exit (0);
62.     }
```

下一篇 《C和指针学习笔记》之指针总结 《C和指针学习笔记》之指针函数的用法