改进多用户系统中的 DOS 盘操作命令

高宏权(工商银行绍兴分行)

摘要:本文就在多用户操作系统下操作 *DOS* 盘上的文件时不能使用通配符的问题进行了分析,并以 *doscp* 为例具体介绍了一种解决方法。

在 Unix、Xenix 等多用户操作系统中,对 DOS 盘上的文件进行操作时,如果在命令中使用了通配符,操作就会挨败。对 DOS 盘上的文件我们只能一个一个地打命令去操作,效率非常低。

在多用户操作系统中,通配符是由命令解释程序 shell 来翻译成相匹配的文件名的。shell 从用户的命令行读到通配符后马上会去搜索文件系统,找出相匹配的文件返回给用户的程序或命令。由于 shell 只能搜索加入本机文件系统的文件,而 DOS 盘的文件管理方法和多用户系统的不一样,不能加入多用户机的文件系统。因此,多用户系统中的 shell 不能搜索 DOS 盘,在多用户操作系统中,如果操作 DOS 盘时使用 shell 认识的通配符,那么必然失败。

虽然多用户系统 shell 不能搜索 DOS 盘,但在多用户系统中还是可以编制程序或命令以 DOS 的文件管理方法来搜索 DOS 盘的(如 doscp dosls dosdir 等)。如果这类命令或程序能从 shell 得到通配符,并在程序内用通配符进行匹配处理,那么这些命令或程序就能在操作 DOS 盘时使用通配符了。

关键问题是,在系统的命令状态,任何应用程序或命令在多数情况下不可能从 shell 那里得到未被翻译的通配符。只有当 shell 搜索不到相匹配的文件时,才把带通配符的参数以原样传递给应用程序或命令。因此,必须用那些没有被多用户系统的 shell 所"认识"的符号作通配符,才能在多用户系统中对 DOS 盘进行操作时使用通配符。

为了使用方便,我对多用户系统中的一系列 DOS 盘操作命令进行了改进。现在我以 doscp 为例谈谈改进的方法。

当用 doscp 从多用户系统拷贝文件到 DOS 盘时,只须搜索多用户系统的文件用不着对 DOS 文件进行搜索,而搜索多用户系的文件时,仍然可以使用原来的通配符"*"和"?",因此不存在对 DOS 盘操作中使用通配符的问题。当用 doscp 从 DOS 盘上拷贝文件时,由于要从 DOS 盘上选择文件,因此就涉及到通配符问题了。

我将改进后的 doscp 改名为 doscpy,在 doscpy 中我选择了多用户系统的 shell 所不认识的"$"和"!"来替代"*"和"?"用作搜索 DOS 盘时的通配符,对于匹配文件名的规则,我还是沿习 DOS 的习惯用法。

1.忽略文件名的大小写。

2.匹配 $ 时:从被搜索文件名的当前字符起直到文件名中出现"."或文件名结束的所有字符都被选择。

3.匹配! 时:被搜索文件名中的当前那个字符被选择。

4.当一个文件名的所有字符都被选择,并且配对参数的所有字符都已处理时,该文件名被选择。

另外,我还使 doscpy 具有了 DOS 中的 shell 所没有的能力,那就是 doscpy 能够处理多个带通配符的参数项。如:doscpy a:$.c $.A!M a!3.$.等。

以下所列的是在 Unix System V 及 Xenix system V 上调试通过的用 C 语言编制的 doscpy 源程序清单:

```
/ * doscpy.c * /
#include <stdio.h>
#include <string.h>
#include <ctype.h>

main (argc,argv)
int argc;
char * argv[ ];
```

```
{
 FILE * pt;
 register int tt,t,i;
 int a,kk,d[550],j,r;
 char cmd[80],ddd[500][14],k[3];

 if(argc<3)
 {
  fprintf(stderr,"用法:doscpy[-rm]device:prth...device:prth\n");
  fprintf(stderr,"通配符替代:$ = * !=?\n");
  exit();
 }
 a=1;
 if(argv[1][0]==='-')
 {if(argv[1][1]!='m'&&argv[1][1]!='r')
   {fprintf(stderr,"参数%c 无效",argv[1][1]);
     exit(1);
   }
   a++;
   sprintf(k,"-%c",argv[1][1]);
 }
e lse sprintf(k,"");
i f(argv[a][1]!=':')
 {
  sprintf(cmd,"doscp");
  for(j=1;j<argc;j++)
  sprintf(cmd,"%s %s",cmd,argv[j]);
  system(cmd);
  exit(0);
 }

 sprintf(cmd,"dosls %c:",argv[a][0]);
 if((pt=popen(cmd,"r"))==NULL) exit(1);

 i=0;
 while(fscanf(pt,"%s",ddd[i]==1)
 {
 j=0;4 while(ddd[i][j]!='\0')
 ddd[i][j]=tolower(ddd[i][j++]);
 i++;
 }
 ddd[--i][0]='\0';
 kk=a;
 r=0;
 if(isspace(argv[a][2])!=0)kk++;
 else r=2;
 for (;kk<argv-1;kk++)
 {
 j=0;
 while(argv[kk][j]!='\\')
 {i=tolower(argv[kk][j]);
   argv[kk][j++]=i;
 }
 j=strlen(argv[kk]);
 i=0;
 while(ddd[i][0]!='\0')
 {
   if(d[i]==1)
   {i++;continue;}
   d[i]=0;
   tt=0;
   for(t=r;t<j;)
   {
    switch(argv[kk][t])
    {
     case '$':while(ddd[i][tt]!='.'&&ddd[i][tt++]!='\0');
        while(argv[kk][t]!='.'&&argv[kk][t++]!='\0');
        break;
     case'!':break;
     default:if(argv[kk][t]!=ddd[i][tt])d[i]=2;
    }
    t++;
    tt++;
    if(d[i]==2)break;
   }
  if(argv[kk][t]=='\0'&&ddd[i][tt]=='\0'&&d[i]!=2)
  {sprintf(cmd,"doscp %s%c:%s %s", k,argv [a][0], ddd[i],
argv [argc-1]);
    system(cmd);
    d[i]=1;
  }
  i++;
  }
 r=0;
 }
}
```