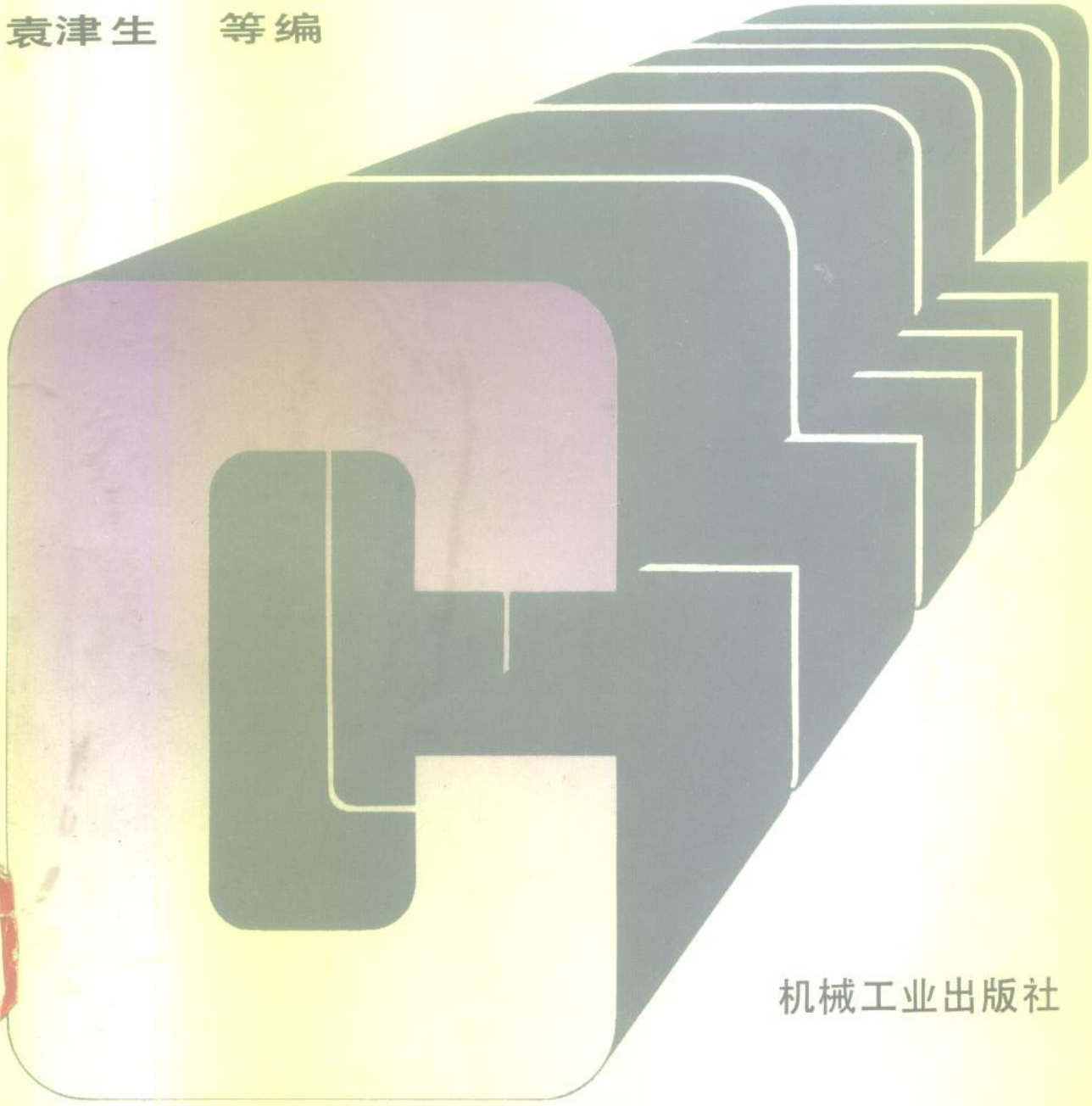
C语言使用技巧及实用程序

袁津生　等编



机械工业出版社

C语言使用技巧及实用程序

袁津生等 编

机械工业出版社

(京)新登字 054 号

本书从实用的角度出发，较全面地介绍了 Turbo C 语言的应用技巧及一些实用程序。全书分为五章，第一章介绍了怎样用 C 语言对 DOS 操作系统的命令进行扩充；第二章介绍了对文件管理的技巧；第三章介绍了音响与图形的应用技巧；第四章介绍了汉字的使用技巧；第五章介绍了各种打印技巧。

本书内容简洁，重点突出。它既可作为 C 语言读者的教材，又可作为 C 语言编程者的使用手册，也可作为 C 语言的子程序库。读者通过对本书的学习，定会提高 C 语言的编程技巧。

图书在版编目(CIP)数据

C 语言使用技巧及实用程序/袁津生等编. —北京：机械工业出版社，1994
ISBN 7-111-04271-9

Ⅰ.C… Ⅱ.袁… Ⅲ.C 语言 Ⅳ.TP312c

中国版本图书馆 CIP 数据核字(94)第 03106 号

出 版 人：马九荣 （北京市百万庄南街 1 号 邮政编码 100037）
责任编辑：王中玉 汪小星 责任校对：肖新民
封面设计：姚 毅 责任印制：卢子祥
北京市密云县印刷厂印刷 · 新华书店北京发行所发行

1994 年 8 月第 1 版 · 1994 年 8 月第 1 次印刷
787mm×1092mm 1/16 · 14.25 印张 · 345 千字
印数 0 001—3 900 册
定价：19.00 元

前　言

C语言自从问世以来就以其强大的功能而成为计算机语言的佼佼者，而Turbo C则以良好的操作环境、丰富的库函数和兼容性好等特点，深受广大用户的欢迎。本书正是以Turbo C2.0为基础，介绍了编者多年来开发应用软件的经验和技巧。

本书不着眼于理论的论述，而着重介绍一些应用技术和使用C语言的技巧。全书共分五章。第一章全面介绍了用C语言来完善DOS命令，完成多任务多功能的技巧，键盘处理技巧以及磁盘管理技巧等。第二章介绍了文件管理技巧，主要内容有文件的读取，文件的保护以及文件加密与解密的技巧等。第三章介绍了菜单技术、图形应用技术和音响技术。第四章介绍了汉字处理技巧。其中有汉字的显示及处理技巧；汉字系统使用及字库的形成技巧；汉字窗口、菜单及一些其它实用程序。第五章介绍了打印技巧。主要内容有文本文件的打印技巧，控制打印机的技巧，图形打印技巧以及自编打印驱动程序的技术等。

书中列举大量实例，并有较详细的注释。所有程序均在Turbo C2.0环境下调试通过，并配备有全部源程序的软盘，购书时可同时购买。本书可作为C语言课程的补充教材；也可作为子程序库使用；也可供有关软件开发者阅读参考。

全书由袁津生、许又生、楚恒林三人编写。由于编者水平有限，书中难免有不少错误和疏漏，欢迎读者和有关专家批评指正。

编　者

1993年10月

目　录

第一章　操作系统使用技巧

微型计算机的 DOS 操作系统是一个具有许多优点的操作系统，目前被人们广泛采用。然而它也有许多不足之处，本章就是针对 DOS 操作系统的不足，介绍采用 C 语言来补充完善的方法。

1.1DOS　命令的完善

1.1.1　完善 TYPE 命令

DOS 的 TYPE 命令允许将文本（ASCII）文件显示到屏幕上。然而此命令没有逐屏显示功能，因此在查看较长的文件时，通常要借助于文本编辑程序。这种方法在查阅大量文件时是很费时的。

用 C 语言编写的程序，除保持 DOS 的 TYPE 命令原风貌外，增加了分屏显示的功能。考虑到与汉字操作系统的兼容性，程序中通过调用 BIOS 功能的中断 INT 10H 来实现字符显示与读取光标现行行。源程序名为 TYPE1.C，经过编译后即可在 DOS 提示符下使用，格式如下：

A〉TYPE1〈文件名〉

TYPE1.C 程序如下：

```
/* 分页显示文件内容 type1.c */
#include " stdio.h"
#include " dos.h"
#include " process.h"
main (argc, argv)
int argc;
char *argv [];
{
FILE *fp;
char ch, *filename;
int row;
union REGS in, out;
if (argc ! =2) {
  printf (" \nUsage: TYPEP filename\n");
  exit (1);
}
filename=argv [1];
fp=fopen (filename," r");
clrscr ();
gotoxy (1, 1);
```

```
row=0;
while ( (ch=fgetc (fp)) ! =EOF) {
  in.h.ah=3 ;
  in.h.bh=0 ;
  int86 (0x10, &in, &out);
  if (out.h.dh ! =row) {
    if (out.h.dh<=23) {
      row=out.h.dh;
      gotoxy (1, row+1);
      } else {
gotoxy (37, 25);
printf (" --more--");
getch ();
clrscr ();
gotoxy (1, 1);
    }
  }
in.h.ah=14;
in.h.al=ch;
in.h.bh=0;
in.h.bl=7;
int86 (0x10, &in, &out);
}
fclose (fp);
}
```

TYPE 命令的另外一个缺点是不能显示非 ASCII 码文件的内容，如. EXE、. COM、. OVL 等文件。但是在大多数这类文件中都含有一些帮助说明、错误提示之类的信息。源程序 TYPE2. C 就可用来读取任意文件内汉字或英文信息。将此程序编译后，在 DOS 下可直接执行，格式如下：

A〉TYPE2〈文件名〉〈参数值〉

其中“参数值”是决定被显示文件中的信息多少而设定的，值越大，显示的信息越少。如：

A〉TYPE2　COMMAND. COM　10

也可利用 DOS 的重定向技术，将显示的信息存入文件中，经编辑后，可打印输出。如：

A〉TYPE2　COMMAND. COM　10〉文件名

程序 TYPE2. C 的设计原理是打开需要显示汉字或英文的文件后，读入字符并且判断该字符是否是可显示字符（0x20 和 0x7e 之间）或者是汉字（大于 0xa1），然后存入数组 str 内，再根据第二个参数来确定是否在屏幕上显示该字符串的信息。

源程序 TYPE2. C 如下：

```
/* 显示任意文件的信息 type2.c */
```

```
#include <stdio.h>
#include <ctype.h>
#include <stdlib.h>
main (argc, argv)
int argc; char *argv [];
{FILE *fp;
char *str; int i, j, k;
if (argc! =3) {
    printf (" usage: c>XS <filename> <parameter>\n");
    exit (1); }
  if ( (fp=fopen (argv [1]," rb")) ==NULL) {
  printf (" Can't open file\n"); exit (1); }
  while (! feof (fp)) {
  k=getc (fp);
  i=0;
  if (isprint (k) || k>0xal) {
  do {
  str [i] =k;
  i++;
  k=getc (fp);}
  while ( (isprint (k) || k>0xal) &&! feof (fp));
  }
  if (i>=atoi (argv [2])) {
  for (j=0; j<i; j++) printf ("%c", str [j]);
  printf (" \n");}
  }
  fclose (fp);
  exit (1);
  }
```

TYPE 命令的又一个缺点是只适应于某种显示行数。如针对 11 行显示环境的显示程序，当用于 25 时，其结果是很大一部分显示屏空着，使屏幕不能充分利用。反之，必将使屏幕上滚，使每页显示信息不能在同一屏上全部看到。

要想使程序具有一定的适应性，就要根据屏幕的显示方式动态地改变其显示行数。程序中是利用 INT 10H 中 0FH 号功能块，来获取当前的显示状态，有了当前的显示状态就可确定当前的显示行数。

另外，用 TYPE 命令显示用 WS 编辑的文本文件时，屏幕上都显示出了软回车符和分页符。为此在程序 TYPE3. C 中，对 WS 编辑的文本文件进行处理，去掉了软回车符和分页符，还对大于 80 个字符的行也做了相应的处理，使超出屏幕的部分自动截至下一行，并且克服了用 TYPE 命令显示而出现半个字符的现象。

程序 TYPE3. C 经过编译后，在 DOS 提示符下使用的格式为：

A〉TYPE3　　〈文件名〉

```
/* 自适应文本文件显示程序 type3.c */
#include <stdio.h>
main (int c1, char *c2 [])
{ FILE *fp, *fopen ();
  int h=0, z=0, j=0;
  char c;
  if (c1<2) {
    printf (" Invalid number of parameter \7\n");
    exit (1);
  }
  if ( (fp=fopen (c2 [1]," rb")) ==NULL) {
    printf (" File not found \7\n");
    exit (2);
  }
for(c=getc(fp);(! feof(fp)&&c! =(char)0xa1);c=getc(fp))
{ if (c>=" ") ++z; /* 空格用区位 0101 输入 */
   if(((++j>=79)&&(z%2==0))|| (c==(char)0x0d)|| (c==(char)0x8d))
   {if((c! =(char)0x0d)&&(c! =(char)0x8d)) printf("%c",c);
   if (h>=mode ())
     { printf (" \7");
   _AH=0x00;
     int86 (0x16); /* 按任一键 */
     printf (" \n"); h=0; z=0; j=0; }
     else
        {if (j>1) printf (" \n"); ++h;
     z=0; j=0;
     }
   }
   else
     { if ( (c! = (char) 0x8a) && (c! = (char) 0x0a))
     printf ("%c", c);
     }
}
fclose (fp);
}
mode () /* 确定显示行数 */
{ _AH=0x0f;
```

```
int86 (0x10);
return ( (_AL==6)? 8 : 22);
}
```

1.1.2 完善 DIR 命令

通常我们用 DIR 命令来查看文件目录，但不能查看隐含的文件和子目录下的文件。为了能查看子目录下的文件和隐含文件，可用 C 语言编写一个程序 DIR1.C，此程序编译后与 DIR 的命令使用相同，只是当被显示的文件是隐含文件时，将在该文件名后标一“*”号。

源程序如下：

```
/* 显示全部文件目录 dir1.c */
#include " dos.h"
#include " dir.h"
#include " string.h"
main (argc, argv)
char *argv [];
{
struct ffblk f;
register int do1, d2;
char path [80] =" \0000", path1 [80] [80], *p;
int level=0, d1=0, d3=0;
path [0] ='A'+getdisk ();
if (argv [1] [1] ==': ') {
path [0] =argv [1] [0];
argv [1] +=2;
}
if ( (p=strchr (argv [1], '*')) ==argv [1]) {
argv [1] +=2;
strupr (argv [1]);
}
strcat (path,":");
do {
if (! (strlen (argv [1]) >0)) printf (" \n%-s\n", path);
d1=0;
strcat (path," \\");
strcpy (path1 [d3], path);
strcat (path," *.*");
do1=findfirst (path, &f, 23);
while (! do1) {
if (stricmp(f.ff_name,".") && stricmp(f.ff_name,"..")) {
if (! stricmp(f.ff_name,argv[1])) {
```

```
printf("\n%s%s%ld",path1[d3],f.ff_name,f.ff_fsize);
}
if ((f.ff_attrib&0x10)==FA_DIREC) {
strcpy (path1 [d3+1],path1[d3]);
strcat (path1[d3++],f.ff_name);
}
if (! strlen (argv [1])) {
printf ("%-s", f.ff_name);
if ( (f.ff_attrib&0x10) ==FA_DIREC) printf (" \\");
if ( (f.ff_attrib&0x02) ==FA_HIDDEN) printf (" *");
else printf ("  ");
for (d2=1; d2< (13-strlen (f.ff_name)); ++d2)
printf ("  ");
if (++d1>5) {
printf (" \n");
d1=0;
}}}
do1=findnext (&f);
}
strcpy (path, path1 [level++]);
} while (level<=d3);
}
```

1.1.3 完善 COPY 命令

DOS 的 COPY 命令可以很方便地把一个盘上的文件拷贝到另一个盘上。但是目前微机上配制的驱动器一般为一个高密驱动器，一个低密驱动器。若要将高密盘上的文件拷贝到低密盘上，光使用 COPY 命令是不行的，因为高密盘上的文件字节数远远超过低密盘所能容纳的字节数。这就必须借助于其它工具软件（如 PCTOOLS）来完成拷贝命令。

我们也可以用 C 语言来编制一个由高密盘到低密盘进行拷贝的程序。其作法是：取出源盘上满足条件的一个文件，与目标盘上可用空间进行比较，若可用空间能容纳源盘上的这个文件，则拷贝这个文件到目标盘，然后取出源盘上满足条件的下一个文件，直到目标盘上的可用空间足够小，然后提示用户换盘。

程序 COPY1.C 经编译后，在 DOS 提示符下使用的格式为

A〉COPY1　A：*.*　　B：　或

A〉COPY1　A：*.PRG　B：　等。

程序 COPY1.C 清单如下：

```
/* 高密盘向低密盘拷贝程序 copy1.c */
#include<stdio.h>
#include<dos.h>
#include<dir.h>
```

```
main (argc, argv)
int argc;
char *argv [];
{
unsigned char driver1, driver2;
int i, done;
long flen;
char order [40], filename1 [20], filename2 [20], c;
struct ffblk f;
struct dfree df;
if (argc! =3)
{
printf (" \7Usage: copy1 a: filename b: \n");
printf (" \nExample: copy1 a: *. prg b: \n");
return (-1);
}
strcpy (filename1, argv [1]);
strcpy (filename2, argv [2]);
printf ("%s is copied to %s\n", filename1, filename2);
done=findfirst (filename1, &f, 0xff);
while (! done)
{
i=1;
while (i==1)
{
getdfree (2, &df);
if (df. df_sclus==0xffff)
return (-1);
flen=f. ff_fsize/1024;
if (df. df_avail>=flen)
{
printf (" copy a: %s b: *. *",f. ff_name);
strcpy (order," copy a:");
strcat (order, f. ff_name);
strcat (order," b:");
strcat (order, f. ff_name);
system (order);
i=0;
}
```

```
else
{
printf (" \7\tThe B: disk is not enough room! Please exchange new disk and then RETURN !" );
c=getchar ();
i=1;
}
}
done=findnext (&f);
}
return (0);
}
```

有时，我们需要将一些大文件拷贝到软盘上去，通常使用 BACKOP 方法。这里介绍的一种方法是将大文件按照软盘的规格截成几个小文件拷入软盘。

拷贝的原理是：根据目前 360KB、1.2MB、720KB、1.44MB 四种规格软盘格式化后可用空间大小，将大文件截断后的小文件长度分别设计成 354KB、1180KB、708KB、1416KB。

例如，将 XSDOS.LPH 截成按 1.2MB 软盘大小装盘，则键入：

C〉COPY2 XSDOS.LPH 2

这样会自动生成 4 个小文件，其中 3 个文件的长度为 1.18MB，可直接拷入软盘中。

源程序 COPY2.C 如下：

```
/* 分割大文件程序 copy2.c */
#include<stdio.h>
#include<io.h>
#include<string.h>
#include<stdlib.h>
#define BK59K     60416L
#define D354K     362496L
#define D1180K    1208320L
#define D708K     724992L
#define D1416K    1449984L
char buf [BK59K];
FILE *fpi;
char fni [10], fno [10], ch, *n, *fnb;

main (argc, argv)
int argc;
char *argv [];
{
FILE * fpo;
ldiv_t lt1, lt2;
```

```
int i, j, fdi, disknb, block, blocknb, remlt;
unsigned long int fli, dsk;
if (argc<2) {
printf (" \007 Usage: DIVIDE <filename> <disktype>\n");
printf (" disktype: 1 - - 360K 2 - - 1.2M\n");
printf ("           3 - - 720K 4 - - 1.44M\n");
exit (0); }
strcpy (fni, argv [1]);
n=strchr (fni, '.');
if (n! =NULL) *n='\0';
strcat (fni,". &");
if ( (fpi==fopen (argv [1]," rb")) ==NULL) {
printf (" Can not open %s ! \n", argv [1]);
exit (1);
}
switch (argv [2] [0]) {
case '1': dsk=D354K; block=6; break;
case '2': dsk=D1180K; block=20; break;
case '3': dsk=D708K; block=12; break;
case '4': dsk=D1416K; block=24; break;
default: break;
}
fdi=fileno (fpi);
fli=filelength (fdi);
lt1=ldiv (fli, dsk);
disknb=lt1.quot;
lt2=ldiv(lt1.rem,BK59K);
blocknb=lt2.quot;
remlt=lt2.rem;
/* write files */
for (j=1; j<=disknb; j++) {
strcpy (fno, fni);
itoa (j, fnb, 10);
strcat (fno, fnb);
fpo=fopen (fno," wb");
/* write disks */
RW (BK59K, fpo, block);
fclose (fpo);
}
```

```
/* write last file less then one disk */
strcpy (fno, fni);
itoa (j, fnb, 10);
strcat (fno, fnb);
fpo=fopen (fno," wb");
RW (BK59K, fpo, blocknb);
/* write rest of bytes less 59K */
RW (remlt, fpo, 1);
fclose (fpi);
fclose (fpo);
}
RW (unsigned long int blk, FILE * fpo, int lop)
{int i;
for (i=0; i<lop; i++) {
fread (buf, sizeof (char), blk, fpi);
fwrite (buf, sizeof (char), blk, fpo);}
}
```

1.1.4 实现磁盘文件的快速搜索技巧

在DOS系统中，文件是通过树形目录结构来管理的。由于硬盘容量大，存储的文件多，加之各用户所建的子目录，要想迅速查找所有目录下的指定文件，只好依次进入各级子目录来搜索，这样做是很费时费力的。

FIND1.C查找各个目录中的指定文件，也可在指定目录下查找某类文件，用法如下：

C〉find1〈文件名〉

```
/* 文件查找程序 find1.c */
#include <stdio.h>
#include <dos.h>
#include <dir.h>
#include <string.h>
#include <ctype.h>
struct ffblk f1;
int i=1, j=0;
char sub_dir [80] [80];
main (argc, argv)
  int argc;
  char *argv [];
{
  int find=0;
  void *p;
  char path [80], name [5], cpath [80], path1 [2] =" C", path2 [2] =":";
```

```
  path1 [0] ='A'+getdisk ();
  if (argv [1] [1] ==':') {
  path1 [0] =argv [1] [0];
  }
  strcat (path1, path2);
  sprintf (path,"%s", path1);
  sprintf (name,"%s"," *. *");
  strcpy (sub_dir [0], path);
  while (1) {
    sprintf (cpath,"%s%s%s", path," \\", name);
    findfirst (cpath, &f1, 0xff);
    do {
      if ( (f1.ff_attrib==0x10)&&(f1.ff_name[0]! ='.'))
{
sprintf(sub_dir[i],"%s\\%s",path,f1.ff_name);
i++;}
} while (! findnext (&f1));
    if (j>=i-1) break;
    strcpy (path, sub_dir [++j]);
      }
clrscr ();
for (j=0; j<i; j++) {
if (argv [1] [1] ==': ') {strnset (argv [1], '\\', 2);
sprintf (cpath,"%s\\%s", sub_dir [j], argv [2]); }
else sprintf (cpath,"%s\\%s", sub_dir [j], argv [1]);
  if (findfirst (cpath, &f1, 0x0f)) continue;
  if (j==0) sprintf (sub_dir [0],"%s\\", path1);
  find=1;
  printf (" \007");
  printf (" Seraching path=%s\n", sub_dir [j]);
  do {
  printf("%20s%20ld\n",f1.ff_name,f1.ff_fsize);
   } while (! findnext (&f1));
  }
  if (! find) printf (" File not found in drive !!!");
}
```

下面的程序可迅速地找到指定的文件及其所在的路径。但是在使用之前，应用环境变量 path 设置相应的路径。算法大致如下：

首先取出环境变量 path 中的内容，并分析 path 的内容，分离出一个个路径名，存入相应