EXE2BIN 命令逆过程的实现方法

铁道部第一勘测设计院西安计算机室 郭永华

摘要:本文分析 *DOS* 两种可运行文件结构特征,介绍 *EXE2BIN* 命令逆过程的实现方法,并给出将.*COM* 文件转换为.*EXE* 文件的实用程序。

笔者通过对.COM 文件和.EXE 文件结构的分析,介绍一种直接转换.COM 文件为.EXE 文件的方法,以实现 EXE2BIN 命令的逆过程,并给出了相应的实用程序,亦可作为 DOS 外部命令的一个补充。

一、具体分析

在 DOS 环境下,.COM 文件和.EXE 文件虽然都是可执行文件,但它们内部结构却不相同。.COM 文件被称为"映象加载"文件,其结构比较简单,文件的内容就是程序本身,当它被调入执行时 IP 自动置 100H;.EXE 文件被称作"段重定位"文件,其结构分为"文件头"和装入模块两部分。"文件头"在文件首部大小是 512B 的整数倍,其中包含 LINK 程序的签字,有效信息为 4D5AH,另外还有重定位项数、装入模块代码、堆栈段的偏移值等(关于.EXE 文件头结构见附表一)。.EXE 文件的装入模块就是程序正文。

二、实现方法

依上所述,笔者设计.COM 文件转换为.EXE 文件的程序基本思路是:将原.COM 文件内容作为.EXE 文件的装入模块部分,以偏移 200H 为起始地址,在其前面空出 512B 留作"文件头",根据.EXE 文件结构要求,填写有关信息生成"文件头"。问题的关键在于,就.EXE 文件而言,在 DOS 加载时其 DS、ES 被置为程序段前缀 PSP 段值,由于 PSP 大小为 1100HB,所以.EXE 文件的开始段值为 DS+10H,而 CS、SS 均为相对段值(由"文件头"给出)加上开始段值;对于.COM 文件,被加载时所有 4 个段寄存器 CS ES DS 和 SS 均指向 PSP+)字段。这样,在正常情况下,若将.COM 文件按照.EXE 文件加载,由于重定位影响会产生+10H 段址偏差。对此,笔者利用地址翻转技术(所谓地址翻转,即段寄存器仅有 16B,若运算时超过 16B,就要产生溢出,地址就发生翻转)纠正了重定位产生的地址偏移,使转换后的.EXE 文件既具有.EXE 文件的标准结构,又能在重定位时获得.COM 文件的内存映象,从而实现了 EXE2BIN 命令的逆过程。

原程序 COM2EXE.C 清单如下,经 Turbo C2.0 编译生成 COM2EXE.EXE 文件。使用时,只需在 COM2EE 后键入待转换.COM 文件名即可。本程序已在 IBM PC / XT 386 等微机上通过,效果理想。

EXE 文件的文件头结构

偏移地址	含 义
00h-01h	EXE 文件有效标志,其内容为 4D5Dh
02h-03h	文件映象长,其值为文件长度除以 512 的余数
04h-05h	文件占用扇区数(包括文件头在内)
06h-07h	重定位表的项数
08h-09h	文件头长度,以节为单位(1 节 = 16 字节)
0Ah-0Bh	加载文件所需的最小字节数
0Ch-0Dh	加载文件所需的最大字节数
0Eh-0Fh	加载模块中堆栈段相对段值
10h-11h	当该程序得到控制时,SP 的值
12h-13h	文件所有字的负累加和
14h-15h	当该程序得到控制时,IP 的值
16h-17h	加载模块中代码段相对段值
18h-19h	重定位表第一个重定位项的位移
1Ah-1Bh	覆盖号(程序驻留为 O)
1Ch…	重定位表
	可变保留区

源程序清单:

```
/ * ……com2exe.c…… * /
/ * Trubo C 2.0 1992.11 * /
/ * ……Guo Yonghua…… * /
#include <io.h>
#include <dos.h>
#include<dir.h>
#include<fcntl.h>
#include<stdio.h>
#include<errno.h>
#include<errno.h>
#define BF64K Oxf000
#define BF480 Ox01e0
#define PER (string)← perror(string);exit(0);}
extern int __fmode, errno, sys__nerr;
extern char * sys__errlist [];
void main(int arge, char * argv[])
{
FILE * fp1, * fp2;
long length;
int i, msize, handle, back;
char drive [MAXDRIVE],dir [MAXDIR],
  file [MAXFILE], ext[MAXEXT];
unsigned char f__head2 [BF480], buf[BF64K],
  out__fname [32];
char * extptr, * p, expt[]="".com"";
char f__head1[32]=
  { Ox4d, Ox5a, Ox00, Ox00, Ox00, Ox00, Ox00, Ox00,
  Ox20, Ox00, Ox00, Ox00 Oxff, Oxff, Oxff, Oxff,
  Oxfe, Ox00, Ox00, Ox00, Ox00, Ox01, Oxf0, Oxff,
  Oxfe, Ox00, Ox00, Ox00, Ox00, Ox00, Oxf0, Oxff,
  Oxff, Ox1e, Ox00, Ox00, Ox00, Ox00, Ox00, Ox00};
if(arge<=1){
  printf("\nUsage:[d:][path]COM2EXE
    input<.COM>[outptu]\n");
  exit(0);}
fnsplit (argv[1],drive,dir,file,ext);
p=expt;
extptr=strlwr(ext);
if(strcmp(p,extptr)!=0){
  printf("\nNot a>COM file!\7");
  exit(1);}
  printf("\nPlease wait …");
fp1=fopen (argv[1],"rb");
  if (fp1==NULL)PER(argv[1])
    handle=fileno (fp1);
    length=filelength(haandle);
  if(length==-1) PER(argv[1])
if(argc!=3){
    fnmerge(out__fname,drive,dir,file,".exe");}
else{strcpy(out__fname,argv[2]);}
fp2=fopen(out__fname,"wb-);
  if(fp2==NULL) PER(out__fname)
    msize=(int)length;
    for(i=0;i<BF480;i++) f__head1[i]=Ox00;
      f__head1 [2]=length%Ox200;
      f__head1[3]=length%Ox200>>8;
      f__head1[4]=(length / Ox200+2);
      f__head1[5]=(length / Ox200+2)>>8;
if(fwrite (f__head1,sizeof(char),Ox20,fp2)!=
  Ox20)PER(out__fname)
if(fwrite(f__head2,sizeof(char), BF480,fp2)!=
  BF480)PER(out__fname)
  back=fread(buf,sizeof(char),msize,fp1);
 if(fwrite(buf,sizeof(char),msize,fp2)!=back)
  PER(out__fname)
   fclose(fp11);
  fclose(fp2);
 printf("\tOK!Transform[%s]into[%s]\n",argv[1],
  out__fname);
```