EXE 可执行文件内存加载过程控制及其应用

葛晓滨　（合肥市统计局计算站）

摘要：本文通过对 *EXE* 文件内存加载过程的分析及控制，提出一种实用的 *EXE* 文件加锁方法，并附程序清单。

一、引言

EXE 文件，是可重定位类型的执行文件。一般的 DOS 用户无需任何复杂手续，便可以启动执行这种程序，不利于软件产权的保护。为此，笔者设计了一个通用的 EXE 文件加锁程序：LOCKEXE，通过控制 EXE 文件的内存加载过程，实现了对 EXE 文件的保护。

二、原理

1.EXE 文件结构的分析

EXE 文件是一种可进行重定位的二进制代码文件，其文件结构由两个部分组成：

a.文件头：包括 EXE 文件的控制和重定位信息。

b.装入模块：包括 EXE 文件的装入字区和可执行代码段。

其文件头的控制信息占 28 个字节的空间，其中包含了 EXE 文件标志、重定位表个数等重要信息，每个信息位均占双字节存储空间，文件头的控制信息结构如图 1 所示：

位移量	内　容	
00H	4DH	5AH
02H	模为 512 的文件长度	
	……	
0AH	装入程序末端需要的最小分配字节数	
0CH	装入程序末端需要的最大分配字节数	
	……	
1AH	由 LINK 产生的覆盖号	

图 *1*　*EXE* 文件头控制信息结构图

2.内存加载过程控制及其加密方法

根据 EXE 文件文件头控制信息的结构，在其文件头位移量 0AH 处起始的两字节信息表示的是文件装入模块末端加载进入内存后所需要申请分配的最小内存量（以十六字节为单位），本文称其为 MiMAB 值（Minimum Memory Allocate Bytes）；在其文件头位移量 0CH 处的两字节信息表示的是文件装入模块末端加载进入内存后所需要申请最大分配的内存量（以十六字节为单位），本文称其为 MaMAB 值（Maximum Memory Allocate Bytes）。对于一般的 EXE 文件，MiMAB 及 MaMAB 值均为 0，使程序尽可能加载装入内存的高地址处。

如果我们将 MiMAB 值设置为接近或等于 FFFFH，则系统会因 EXE 文件所需要申请最小分配内存最太大而发出警告："Program too big to fit in memory"，同时退出 EXE 文件的执行。

利用 MiMAB 控制 EXE 文件加载进入内存的这一功用，我们将 EXE 文件的 MiMAB 采用算法：SMiMAB= MiMAB ⊕ FFFFH，将 MiMAB 改变为接近或等于 FFFFH 的数值 SMiMAB（其中，符号 ⊕ 代表异或运算），经过这样处理的 EXE 文件，由于在申请内存量上的限制而无法运行，文件即被加锁。同理，加锁 EXE 文件的解锁算法只需将 SMiMAB 与 FFFFH 再次进行异或，即采用算法：MiMAB= SMiMAB ⊕ FFFFH，便可将 SMiMAB 还原为 MiMAB，使 EXE 文件可恢复使用。

三、实现程序

根据上述原理和方法，用 C 语言设计了一个通用 EXE 文件加锁软件：LOCKEXE.C，它可以对用户指定目录下的 EXE 文件进行加锁（解锁），其运行格式如下：

lockexe [-a] file1 [file2]…

其中[-a]参数表明对磁盘上所有目录下的 EXE 文件均进行加锁（解锁）操作，file1、file2 等是用户指明的待

© 中国科学院软件研究所 http://www.c-s-a.org.cn

加锁(解锁)的文件名。文件名可以带驱动器名及路径名,并可以根据用户需要一次指定多个文件进行加锁(解锁),若指定多个文件,则文件名间以空格作为分隔符,文件名还可以根据用户需要使用 DOS 的通配符"?"及"*"。举例说明如下:

a. C>lockexe -a *.exe

b. C>lockexe s*.exe m*.exe

格式 a 是将硬盘上所有目录下的 EXE 文件加锁(解锁);格式 b 是将硬盘上当前目录下的以字母"S"开头以及"M"开头的 EXE 文件加锁(解锁)。

LOCKEXE 程序的特点是运行奇数次是对 EXE 文件加锁,运行偶数次是对 EXE 文件解锁。

四、应用

LOCKEXE 为 DOS 用户保护 EXE 文件提供了方便。例如,LOCKEXE 可以一次性将 EXE 硬盘上所有的 EXE 文件全部加锁保护,其它的用户即使从硬盘上拷贝到这种 EXE 文件,也无法使用。而 LOCKEXE 的持有者可以将 LOCKEXE.EXE 软件独立存放在自己的软盘中,当其需要使用硬盘上的 EXE 加锁文件时,只需再运行一次软盘上的 LOCKEXE 程序即可将硬盘上所有加锁的 EXE 程序解锁使用。

附:程序清单

```
/*程序名:LOCKEXE.C*/
/*编程语言:Turbo C 2.0*/
#include "stdio.h"
#include "dir.h"
#include "ctype.h"
#include "string.h"
main(int argc,char *argv[ ]){
 int a,b,i,j,k,m,c[2],d[2],e[2];
 char *file[20];
 struct ffblk f;
 FILE *fp;
 clrscr();
 printf("\n\t\t<<<LOCKEXE>>>");
 printf("\n\t Author: X.B.GE");
 printf("\n\t Usage: LOCKEXE [-a] file1 [file2] ...");
 printf("\n");
 if(argc<2){
  printf("\n\t Warning : Bad usage !");
  exit(1);
 if(!strcmp((char *) strlwr((char *) argv[1],"-a"))
  a=2;
  else
  a=1;
 while(argv[a]){
  b=findfirst(argv[a],&f,0);
  if(b==-1){
   printf("\n\t Can't open file %14s", argv[a]);
   goto SKIPNEXT;
 }
 i=0;
 while (!b){
  strcpy(file[i],f.ff_name);
  i++;
  b=findnext(&f);
 }
 for(j=0;j<i;j++){
  fp=fopen(file[j],"rb+");
  if(fp==NULL){
   printf("\n\t File error in :%14s!", file[j]);
   goto SKIPNEXT;
  }
  rewind(fp);
  for(m=0;m<=1;m++)
   e[m]=fgetc(fp);
  if(e[0]!=0x4d      e[1]!=0x5a){
   printf("\n\t %14s isn't EXE File!", file[j]);
   goto SKIPNEXT;
  }
  rewind(fp);
  fseek(fp,10,SEEK_SET);
  for(k=0;k<2;k++){
   c[k]=fgetc(fp);
   d[k]=c[k]^0xff;
  }
  rewind(fp);
  fseek(fp,10,SEEK_SET);
  for(k=0;k<2;k++)
   fputc(d[k],fp);
  fclose(fp);
  }
  SKIPNEXT: a++;
  }
  printf("\n\tFile(s) is success LOCKed / UNLOCKed!");
  printf("\n\t\t Press any to exit!");
  getch( );
  clrscr( );
}
```


© 中国科学院软件研究所 http://www.c-s-a.org.cn