

HOPE

Microsoft OS/2 程序设计指导与实例

夏洪山 等编译

(共二册) 下册



中国科学院希望高级电脑技术公司

HOPE

Microsoft OS / 2

程序设计指导与实例

(下册)

夏洪山 等编译

中国科学院希望高级电脑技术公司

一九九一年元月

编 译 者 的 话

Microsoft OS/2 的问世,给广大PC用户和软件工作者带来福音:没有640KRAM障碍,支持单用户多作业。

然而,与PC DOS下的程序设计相比较,为OS/2编程则要复杂一些。为此,本书从实用角度出发,在介绍OS/2概念和内部运行原理的同时,给出许多短小的程序例子,帮助说明某一概念。本书要详细介绍OS/2的屏幕显示,文件操作,多道作业,虚拟存贮管理(内存的共享以及保护),进程通讯和并发函数(队列,管道和信号)等功能,以及OS/2的API函数。如何避免和解决编程中容易发生的问题,诸如多道作业的潜在失控,资源竞争以及死锁等,本书也做了详尽的阐述。

本书既是OS/2的入门,也是OS/2程序设计的手册。

参加本书编译工作的主要有:夏洪山、仇钟、邱文宁、丁丽亚、薛冰、汪瑾和宋震等人,在完成本书的工作过程中,得到了阎仲平、姚小康以及有关部门的鼎力相助,中科院希望电脑公司对本书的印刷工作进行了认真细致的组织安排,值此一致表示衷心感谢!

限于编译者的水平,书中谬误难免,敬请读者批评指正。

编 译 者

一九九〇年六月十日

目 录

第一章 OS/2概述

- 1.1 操作系统的简要历史..... (5)
- 1.2 批处理系统..... (5)
- 1.3 OS/2 特性..... (9)
- 1.4 程序员如何使用OS/2 (16)
- 1.5 本章概要..... (18)

第二章 准备过程

- 2.1 建立OS/2 环境..... (18)
- 2.2 简单的OS/2 程序..... (23)
- 2.3 本章概要..... (36)

第三章 键盘与显示屏幕

- 3.1 写屏幕..... (37)
- 3.2 光标 (42)
- 3.3 读屏幕..... (47)
- 3.4 滚行 (48)
- 3.5 键盘 (50)
- 3.6 视频模式..... (57)
- 3.7 本章概要..... (65)

第四章 进程

- 4.1 一父一子的进程..... (66)
- 4.2 多子进程的父进程..... (70)
- 4.3 获悉子进程何时结束..... (72)
- 4.4 向子进程传送参数..... (75)
- 4.5 父进程与孪生进程..... (77)
- 4.6 优先级..... (81)
- 4.7 后台进程..... (83)
- 4.8 关于进程的进一步思考..... (87)
- 4.9 小结 (88)

第五章 并发函数

- 5.1 创建并发函数 (88)
- 5.2 从单个并发函数中退出 (93)
- 5.3 局部和全局变量 (95)
- 5.4 从多个并发函数中访问同一个函数 (96)
- 5.5 共享代码..... (101)

5.6	挂起并发函数.....	(106)
5.7	中途退出.....	(108)
5.8	并发函数应用的例子.....	(111)
5.9	小结.....	(119)
第六章 并发信号		
6.1	进程间通信协调问题.....	(119)
6.2	并发信号标志.....	(122)
6.3	并发函数之间发信操作.....	(131)
6.4	系统并发信号标志.....	(143)
6.5	死锁.....	(154)
6.6	小结.....	(157)
第七章 文件目录		
7.1	目录操作.....	(158)
7.2	文件操作.....	(163)
7.3	查找文件.....	(167)
7.4	文件属性.....	(178)
7.5	查找一个已知文件的路径.....	(181)
7.6	小结.....	(184)
第八章 文件		
8.1	文件的读和写.....	(185)
8.2	文件信息.....	(196)
8.3	其它文件函数.....	(203)
8.4	总结.....	(207)
第九章 文件与多道作业		
9.1	同时读和分析.....	(208)
9.2	同时读和显示.....	(222)
9.3	进程间共享文件.....	(226)
9.4	总结.....	(236)
第十章 内存管理与操作		
10.1	为什么分配内存.....	(236)
10.2	MS-DOS和OS/2 的区别.....	(237)
10.3	内存和硬件.....	(237)
10.4	简单的段分配.....	(241)
10.5	分配多重段.....	(246)
10.6	虚拟内存.....	(248)
10.7	内存分配.....	(250)
10.8	巨型目标.....	(252)
10.9	改变段的大小.....	(259)

10.10	段中再分配的内存块	(262)
10.11	丢弃与加锁的段	(266)
10.12	代码段的别名	(268)
10.13	小结	(270)
第十一章	进程间的通讯	
11.1	什么时候使用什么	(271)
11.2	共享内存	(272)
11.3	管道	(283)
11.4	队列	(291)
11.5	小结	(307)
第十二章	监控程序与信号	
12.1	设备监控器	(308)
12.2	TRS的 OS/2 方法	(308)
12.3	监控器与多任务	(309)
12.4	键盘输入的数据路径	(309)
12.5	监控器应用程序的组织	(310)
12.6	读/写数据包	(310)
12.7	删除监控器	(311)
12.8	一个简单的监控器应用程序	(312)
12.9	Kbdmon.h头文件	(314)
12.10	键盘输入包	(314)
12.11	monflc程序的操作	(316)
12.12	给数据流增加若干包: montime.c	(319)
12.13	危险与责任	(325)
12.14	信号	(325)
12.15	小结	(332)
第十三章	鼠标与图形	
13.1	鼠标	(332)
13.2	图形	(348)
13.3	小结	(362)
第十四章	动态链接	
14.1	动态链接基础	(362)
14.2	装入期动态链接	(367)
14.3	运行期链接	(372)
14.4	例子与全程数据	(375)
14.5	初始化例程	(380)
14.6	小结	(383)

附录A OS/2函数	(384)
A1 引言	(384)
A2 OS/2 函数	(384)
附录B 输入和输出控制函数	(607)
B1 引言	(607)
B2 各种分类码和函数码	(608)
B3 物理磁盘控制	(608)
B4 分类码和函数码	(608)
B5 函数	(611)
附录C OS/2宏	(679)
C1 引言	(679)
C2 宏调用	(680)
附录D 视屏模式	(682)
D1 引言	(682)
D2 屏幕模式	(683)
D3 屏幕属性	(683)
D4 物理屏幕缓冲区地址	(684)
附录E OS/2键盘转换表	(684)
E1 引言	(684)
E2 预定义的转换表	(684)
E3 转换表格式	(685)
E4 键盘型	(687)
附录F OS/2可执行文件格式	(697)
F1 引言	(697)
F2 MS-DOS 可执行文件头部	(698)
F3 新的可执行文件头部	(698)
F4 段表	(699)
F5 源表	(700)
F6 模块调用表	(701)
F7 入口点表	(701)
F8 驻留与非驻留名字表	(702)
F9 输入名表	(702)
F10 段	(702)
附录G OS/2下的ANSI.SYS	(703)
G1 引言	(703)
G2 光标函数	(704)
G3 擦除函数	(704)
G4 屏幕图形函数	(704)

附录A OS/2 函数

1 引言

本章将按字母的顺序列出MSOS/2函数。并列举一些简单的程序例子来说明如何使用某一指定函数。

2 OS/2函数

本节列出MS OS/2系统函数。并说明每个函数的用途,语法、参数和可能有的返回值。

USHORT DosAllocHuge(usNumSeg,usPartialSeg,psel,usMaxNumSeg,fAlloc)

USHORT usNumSeg, 65, 536字节段的数目

USHORT uspartialSeg; 最后一个段的字节数

PSEL psel; 指向由选择器所分析的变量指针

USHORT usMaxNumSeg; 65, 536字节段的最大度

USHORT fAlloc; 分析\废弃标志

DosAllocHuge函数分配一个巨型内存块。一个巨型内存块由一个或多个内存段组成,每个段有65, 536个字节,另外一个附加段有usPartialSeg个字节。若usPartialSeg参数为0,则不分配附加段。

DosAllocHage函数分配段,并将第一个段的选择器复制进由psel参数所指定的变量中。剩余段的选择器是连续的,并且必须使用选择器偏移量来计算。使用DosGiveSeg函数可以使进程所创建的段与其它进程共享。

DosAllocHuge函数是一个API函数。

参数 说明

usNumsge	指定所分配的65, 536字节的段数。
uspartialSeg	指定在最后一个段中的字节数。可以从0到65, 536中的任一个数值。若为0,则不分配附加段。
psel	指向接收第一个段选择器的变量。
usMaxNumSeg	指定再分配的最大量。这是在以任何顺序调用DocAllocHuge函数所能指定的最多段数。若usMaxNumSeg参数为0,则再分配的内存不能大于它原有尺寸,但它可以与原有尺寸相等。
fAlloc	指定段是否可以为其它进程所共享或可被废弃。 fAlloc参数为下列形式之一:
值	含义
0X0000	创建一个不可共享、不可丢弃的段。
SEG-GIVEABLE	通过调用Dos GiveSeg函数创建的、属主进程可以给它给予其它进程的共享段。
SEG-GETABLE	通过调用Dos Give Seg函数创建的一个其它进程可以读取数据的共享段。
SEG-DISCARDABLE	创建一个可丢弃段。

返回值

若调用函数成功, 返回值为 0。否则返回一个错误值, 其值为下列值:

ERROR—NOT—ENOUGH—MEMORY

注释:

在巨型内存块中的每一个段数具有唯一的选择器。选择器是连续的, `psel` 参数指定第一个选择器的值。可以通过一次或多次将选择器的偏移量加到第一个选择器上来计算剩余选择器, 即一次为第 2, 二次为第 3, 等等。选择器偏移量是调用 `DosGetHugeShift` 函数所取得的移位量的 2 倍。例如, 若移位位数为 2, 则选择器偏移量为 4 (即 1 左移 2 位等于 4)。若选择器偏移量为 4, 第一个选择器为 6, 那么第二个选择器为 10, 第三个为 14, 等等。

系统可以按照 `config.sys` 文件中 `memman` 命令所指出的那样移动或转贮内存段。移动或转贮不影响段选择器的值, 因此在任何时候, 都可以计算选择器并存储它们; 只要分配不被收回, 它们将一直能使用。

若被放弃它的第一个选择器, 则 `DosFreeSeg` 函数释放所有的段。

API 类函数的限制:

在实模式中, `DosAllocHuge` 函数受如下限制:

- `usPartialSef` 参数被循环到下一段值 (16 字节)。

- 实际地段地址写入 `psel` 参数。

举例:

本例调用 `DosAllocHuge` 函数, 分配两个 64K 段和一个 200 字节的段。然后它将第一选择器转换成可以访问所有被分配内存的一个远指针:

```
CHAR huee *pobBuffer;  
SEL sel;  
DosAllocHuge(3, /*number of segment  
200, /*Size of last segment  
sel, /*address of selector  
0, /*sharing flag  
5); /*maximum Segment for reallo  
pchEuffer=MAKEP(sel, 0) /*Conuerts to a pojnter
```

参见

`DosFreeSeg`, `DosGetHugeShift`, `DosGetSeg`, `DosGiveSeg`.

`DosReallocHuge`

USHORT DosAllocSeg(usSize,pselAlloc)

USHORT usSize; 所申请的字节数

PSEL Psel 接收所分配的选择器

USHORT falloc 分配标志

`DosAllocSeg` 函数分配一个内存段并且将段选择器写入由 `psel` 参数所指向的变量。段可以从 1 到 65,536 个字节, 具体大小由 `usSize` 参数未定义。

当DosAllocSeg分配一个段时,它使用由fAlloc参数所定义的属性。段可以共享或丢弃。共享段可以用在创建它的进程或其它进程中。调用DosGiveSeg和DosGetSeg函数可以使进程共享一个共享段。丢弃段是系统可丢弃的段,以便满足一些其它进程所申请的分配。丢弃段作为保持可以被迅速更新的临时数据是很有用的。若不定义共享或丢弃属性,则只能由创建该段的进程来访问它,并在进程释放段以前,它的内容一直保留在内存中。

DosAllocSeg函数是API类函数。

参数	说 明										
us Size	指定所分配的字节数。可以是0到65,535之间的任何值。若其为0,则函数分配,65,535个字节。										
Psel	指向接收段选择器的变量。										
fAlloc	指出段的分配标志。可以是下列值的任何组合: <table border="1"> <thead> <tr> <th>值</th><th>含义</th></tr> </thead> <tbody> <tr> <td>0x0000</td><td>创建一个不可共享段。不可丢弃段。</td></tr> <tr> <td>SEG-GINEABLE</td><td>通过调用DosGiveSeg函数可以使拥有创建共享段的进程将共享段给其它进程共享。</td></tr> <tr> <td>SEG-GETTABLE</td><td>通过调用DosGetSeg函数创建一个其它进程可以取回的共享段。</td></tr> <tr> <td>SEG-DISCARDABLE</td><td>创建一个丢弃段。</td></tr> </tbody> </table>	值	含义	0x0000	创建一个不可共享段。不可丢弃段。	SEG-GINEABLE	通过调用DosGiveSeg函数可以使拥有创建共享段的进程将共享段给其它进程共享。	SEG-GETTABLE	通过调用DosGetSeg函数创建一个其它进程可以取回的共享段。	SEG-DISCARDABLE	创建一个丢弃段。
值	含义										
0x0000	创建一个不可共享段。不可丢弃段。										
SEG-GINEABLE	通过调用DosGiveSeg函数可以使拥有创建共享段的进程将共享段给其它进程共享。										
SEG-GETTABLE	通过调用DosGetSeg函数创建一个其它进程可以取回的共享段。										
SEG-DISCARDABLE	创建一个丢弃段。										

返回值

若函数调用成功,返回值为0。否则,它为一出错值,可以为如下形式:
ERROR—NOT—ENOUGH—MEMORY

注释

系统可以象由config.sys文件中的memman命令所指示的那样移动或转贮内存段。移动或转贮不影响段选择器的值,因此在任何时候都可以计算选择器并保存它们;只要分配可被收回,它们将一直保持可用。

对于保持期内可以访问的信息,和迅速恢复所丢弃的信息,丢弃段是很有用的。例如,用作数据库程序包,高速缓冲存储器的缓冲区,用作保存窗口位图,以及用作字处理应用预先计算的显示用图像。若一个段是可丢弃段,存储器管理程序可以丢弃它,如果系统在存储器的低部。尽管在段中的数据被丢失,通过调用DosReallocSeg函数可以将段恢复到它的起始尺寸。为了防止内存管理程序丢弃段,可以使用DosLockSeg函数来锁存之。一个锁存的段不能被丢弃。DosUnlockSeg函数对段解锁并允许丢弃。DosAllocSeg和DosReallocSeg函数自动锁存段。

DosFreeSeg函数释放段。

API类限制:

在实模式中,DosAllocSeg受如下限制:

- unsize参数舍入到下一个段(16字节)。
- 实际段地址写入psel参数。

举例:

本例调用DosAllocSeg函数来分配26,953个字节。然后将选择器转换成一个可以访问所分配的字节的远指针。

```
PCH pcbBuffer;  
SEL sel;  
DOS AllocSeg(26953,          /* byte to allocate  
    &sel,                    /* address of selector  
    0);                      /* sharng flag  
pch Buffer=MAKEP(SEL, 0) /* Converts to a pointer
```

参见

DosFreeSeg, DosGletSeg,
DosLockSeg, DosReallocSeg, DosUnlockSeg
USHhRT DosAllocShrSeg(usSize, pszSegName, psel)

USHORT usSize; 所申请的字节数。
PSI pszSegName; 指向段名的指针。
PSEL psel; 指向所分配的段选择器变量指针。

DosAllocShrSeg函数分配一个共享内存段,并且将段选择器写入由psel参数所指向的变量中。段可以有1到65,536个字节,具体大小由usSize参数来定义。

一个共享段可以由知道它的名字的任何进程访问。通过在调用DosGetShrSeg函数中指定的段名,进程可以取得段选择器。使用DosAllocSeg函数所分配的共享段是不相同的。使用DosAllocSeg所分配的段必须通过调用DosGiveSeg和DosGetSeg函数明确给出或取出。

参数	说 明
----	-----

usSize	指出所分配的字节数。可以是0到65,535之间的任何一个值。若为0,则函数分配65,535个字节。
--------	---

pszSegName	指向一个以空终止的字符串。该串标识共享内存块,并且必须为如下格式: /sharemem/name 段名name必须与MSOS/2文件名具有相同的格式,必须是唯一的。例如名字\sharemem/public.dat是可以接受的。
------------	---

psel	指向接收段选择器的变量。
------	--------------

返回值

若调用函数成功,返回值为0。否则,它为一个出错信息。出错信息可以是下列之一:

ERROR—ALREADY—EXISTS

共享段已经存在。

ERROR—INVALIDHANDLE

共享段名无效。

ERROR—NOT—ENOUGH—MEMORY

内存不够。

注释:

通过使用DosGetShrSeg函数和指定的段名,任何其它进程都能获取一个共享段的

选择器。尽管共享内存选择器对于各个进程来说可能不尽相同，但它们确实标识着同一内存。

一个进程最多可以分配30个共享段。

系统可以象config.sys文件中的memman命令所指定的那样移动或交换内存段。移动或交换内存段不影响段选择器的值。

DosFreeSeg函数释放一个共享段。

举例：

本例调用DosAllocShrSeg函数分配26,953个字节。它给内存的名字为“\SHAREMEM\ABC.MEM”，若其它进程知道这个名字，它们也可以使用该内存：

```
SEL sel
DosAllocShrSeg ( 26953,          /* 分配的字节 */
"\\SHAREMEM\\ABC.MEM" /* 内存名 */
&sel )          /* 选择器地址 */C
```

参见

DosAllocHuge, DosAllocSeg, DosFreeSeg, DosGetShrSeg。

USHORT DosBeep; 频率，以Hz为单位的频率；

USHORT usDuration; 以毫秒为单位的持续时间。

DosBeep函数使扬声器发出声响。

DosBeep函数是API类函数。

参数 说明

usFrequency 指出以Hz为单位的频率（周期/秒），它可以是以0x00025到0x7FFF的任意值。

usDuration 指出以毫秒为单位的持续时间。

返回值

若函数调用成功，返回值为0。否则它为一出错信息。其值可以是如收形式：

ERROR-INVALID-FREQUENCY

频率超出0x00025到0x7FFF的范围。

举例：

本例调用DosBeep函数产生一个可以听见的音响：

```
int i;
for ( i=0; i<10; i++ ) (
    DosBeep ( 600, 175 );
    DosBeep ( 1200, 175 );
```

USHORT DosBufReset(hf)

HFILE hf; 文件把柄

DosBufReset函数刷新指定文件的文件缓冲区。它通过将当前文件缓冲区的内容写入相应设备来刷新缓冲区（例如，作为磁盘文件写入磁盘）。若文件为磁盘文件，则函数还将修改文件的目录信息。

尽管DosBufReset函数刷新和修改信息看起来文件象是关闭了，但实际上文件还是打

升着的。美斯利尖前个官月，同时零环前河前米盛个个各于该器有数容内享其管以。器科数3。

DosBufResef函数是API类函数。

参数 说明

hf 指出缓冲区被刷新的文件。把柄必须已经使用Dos Open函数预先创建。若hf参数设置为0 X EFFF，则函数刷新所有当前打开的文件缓冲区。

返回值

若调用函数成功，返回值为0。否则返回一出错信息。其值为下列形式之一：

ERROR—ACCESS—DENIED；产生一个磁盘错误。

ERROR—FILE—NOT—FOUND；文件未找到。

ERROR—INVALID—HANDLE；无效把柄。

注释

若进程在软盘上有几个打开文件，则该函数可能会导致需要用户倒盘许多次。

举例

本例打开文件abc，并将abBuf的内容写入文件。然后调用DosBufReset函数来刷新缓冲区。以便在文件关闭前将数据写入磁盘：

```
BYTE abBuf[ 512 ];
HFILE hf;
USHORT usAction, cbBytesWritten;
if (! DosOpen ( "abc", &hf, &usAction, OL, 0, 0x10, 0x41, OL ) ) { DosWrite ( hf, abBuf,
sizeof ( abBuf ), &cbBytesWritten );
DosBufReset(hf);
```

参见：

DosClose, DosOpen, DosWrite

USHORT DosCaseMap(vsLength, pctrycCountry, Pch String)

USHORT usLngth; 大小写匹配的字符串长度

PCOUNTRYCODE pctrycCountry; 指向COUNTRYCODE结构；

PCHAR pchString; 指向字符串；

DosCaseMap函数在所给定的字符串中进行字符的大小写匹配。若需要，函数用给出的字符串中所存在的字符来替换当前匹配的字符。

DosCaseMap函数是API类函数。

参数 说明

usLength 指出所给字符串的长度。

pctrycCountry 指向一个COUNTRYCODE结构，该结构包含标识大小写匹配操作的country码和代码页。有关详细说明，参见下面“结构”部分。

pchstring 指向被匹配的字符串。

返回值

若调用函数成功，返回值为0。否则返回一个出错值。其值可以是下列形式之一：

ERROR—NLS—BAD—TYPE

产生一个内部错误，选择器类型不存在。

ERROR-NLS-NO-COUNTRY-FILE

找不到country.sys文件。

ERROR-NLS-NO-CTRY-CODE

在country.sys文件中找不到国别代码。

ERROR-NLS-OPEN-FAILED

不能打开country.sys文件。

ERROR-NLS-TABLE-TRUNCATED

缓冲区太小。

ERROR-NLS-TYPE-NOT-FOUND

产生一个内部错误：选振器类型不在文件中。

结构

由pctrycCountry参数所指向的COUNTRYCODE结构的格式如下：

```
typedef struct COUNTRYCODE {  
    USTORT country;  
    USTORT codepage;  
} COUNTRYCODE;
```

字段

说明	字段
指出用于大小写匹配的国别代码。若为0，函数使用为前国别代码。函数返回时，country字段接收函数实际用于大小写匹配的国别代码。	country
指向代码页标志。它用于确定使用什么大小写匹配信息。	codePage

注释：

DosCaseMap函数使用country.sys文件中的大小写匹配信息来匹配字符串。

API类的限制：

在实模式中，DosCaseMap函数受如下限制：

- 没有识别引导驱动器的方法。系统假定country.sys文件在当前驱动器的根目录下。

参见

DosGetCollate, DosGetCtryInfo, DosSetCp

USHORT DosChdir (pszDirpath, ulReserved)

PSZ pszDirPath; 目录路径

ULONG ulReserved; 必须是0

DosChdir函数将当前目录改变成指定的目录。当一个进程改变当前目录时，随后调用文件系统函数，如DosOpen函数，使用新目录作为默认目录。这意味着若没有给出一个带文件名的明确路径，则那些函数使用新目录。

DosChdir函数是API类函数。

参数

说明	参数
指向一个确定新目录路径以空字符终止的字符串。该字符串必须是一个正确的MSDOS/2 目录路径名并且长度不超过64个字符。	pszDirpath
指定一个保留值，该值必须为0。	ulReserved

返回值

若调用函数成功, 返回值为 0。否则, 它返回一个出错信息。其值可以是下列形式之

一:

ERROR—DRIVE—LOCKED;
ERROR—FILE—NOT—FOUND;
ERROR—NOT—DOS—DISE;
ERROR—NOT—ENOUGH—MEMORY;
ERROR—PATH—NOT—FOUNT

注释:

该函数只适用于进程改变目录。它并不影响其它进程的当前目录。

当一个进程开始时, 它继承来自其父进程的当前目录。

举例

该例保存当前默认驱动器和路径, 然后调用DosChdir函数将默认路径变为根目录:

```
CHAR SzDirpath[ 255 ];          /* buffer for path name */
USHORT cbDirpath=sizeof(syDirpath), usDisk,
ULONG ulLogicalDrives;
DosQCurDisk(&usDisk, &ulLogicalDrives);    /* Gets current dlive */
DosQCurDir(usDiSk, szDirpath, &cbDirpath)  /* current directooy */
DoSCgdir("/ ", OL);                /* Changes totherootdirectory */
.
.
.
DosChdir(szDirpath, OL);           /* Restores the direcsory */
```

参见

Dos Mkdir, DosQCvrDir, DosQCurDisk,
DosRmdir, DosSelectDisk,

USHORT DosChgFileptr(hf, lDistance, fMethod, pulNewptr)

HFILE hf;	文件把柄
LoUNG lDistance;	移动距离
USHORT fMethod;	移动方式 (0, 1, 2)
PULONG ptlNewptr;	新指针位置

DosChgFileptr函数在文件中将文件指针移到一个新的位置。由系统保持的文件指针指向从文件中所读取的下一个字节或在文件中的下一个位置来写一个字节。DosChgFileptr将指针移到一个所要求的位置, 该位置是相对于文件的开始还是相对于文件的结束, 还是相对于当前位置, 均由fMethod参数确定。指针所移动的距离和方向取决于lDistance参数的值和符号。

DosChgFileptr函数是API类函数。

参数 说明

hf

标识文件。必须预先调用。DosOpen函数创建文件把柄,

LDistance	指出在文件中移动文件指针的字节数。若为正, 在文件中指针始终向正方向移动(由开始到结束)。若为负, 指针则由结束方向向开始方向移动。
hlfMethod	指出从何处开始移动。它必须为下列值之一:
值	含义
0x00000	从文件的开始处移动。
0x00001	从当前位置处移动。
0x00002	从文件的结束处移动。
pulNewPtr	指向接收新的文件指针位置的长变量。

返回值

若调用函数成功, 返回值为 0。否则, 它返回一个出错值。该值可以为下列值之一:
ERROR-INVALID-FUNCTION,
 非法 fMethod 参数。

ERROR-INVALID-HANDLE.

注释

系统对于每个字节的读或写自动将文件指针前移, 当文件打开时, 指针指向文件的开始处。

举例

本例为了读和写访问而打开文件 abc, 调用 DosChgFileptr 函数将文件指针设置在文件的结尾, 写出字符串 "Hello, World", 然后关闭文件。当指针设置在文件的结束处时, ulFilepointer 变量含有文件的当前长度:

```

HFILE hf;
USHORT usAction, cbBytesWritten;
ULONG ulFilePointer;
DosOpen("abc", &hf, &usAction, OL, 0, 0x11, 0x42, OL);
DosChgFilePtr(hf, /* file handle */
               OL, /* distance to move */
               2, /* type of movement */
               &ulFilePointer) /* address of new position */
DosWrite(hf, "Hello World\n\r", 13, &cbBytesWritten);
DosClose(hf);

```

参见

DosNewSize, DosOpen, DosRead, Doswrite

USHORT DosCLIAccess(VOID)

为了禁止或使能中断, DosCLIAccess 函数申请一个输入/输出(I/O)优先权。在 IOPL 段中使用 cli 和 sti 指令的汇编语言程序必须使用 DosCLIAccess 函数来接收一个使用这些指令的许可。

DosCLIAccess 函数是一个 API 类函数。

该函数无参数。

返回值

若调用函数成功, 返回值为 0。否则返回一个出错值。

注释

想要使用 in 和 out 指令读和写 I/O 口的汇编语言程序, 必须使用 DosPortAccess 函数来接收使用这些指令的许可。DosPortAccess 函数还接受使用 cli 和 sti 指令的许可。

参见

DosPortAccess

USHORT DosClose (hf)

HFILE hf, 文件把柄

DosClose 函数关闭由 hf 参数指定的文件。DosClose 函数控制系统将作为文件的所有内部缓冲区的内容写入记录媒介中 (例如, 磁盘) 然后修改所有的目录信息。

DosClose 函数是一个 API 类函数。

参数 说明

hf 标识被关闭的文件。必须使用 DosOpen 函数预先创建文件把柄。

返回值

若调用函数成功, 返回值为 0。否则, 它为一出错信息。出错信息可以为下列信息之一:

ERROR—ACCESS—DENIED;

产生一个磁盘错误;

ERROR—FILE—NOT—FOUND

ERROR—INVALID—HANDLE

举例

本例打开文件 abc, 读入 512 个字节并调用 DosClose 函数关闭文件:

```
BYTE abBuf[512];
HFILE hf;
USHORT usAction, cbBytesRead;
DosOpen("abc", &hf, &usAction, OL, 0, 0x01, 0x42, OL);
DosRead(hf, abBuf, sizeof(abBuf), &cbBytesRead);
DosClose(hf); /* Closes the file */
```

参见

DosBufReset, DosFindClose, DosOpen

USHORT DosCloseQueue(hqueue)

HQUEUE hqueue; 队列把柄

DosCloseQueue 函数关闭一个队列。若调用 DosCloseQueue 的进程拥有队列, 则函数去除任何在队列外面的元素。若进程不拥有它, 则队列内容保持不变, 其它可以使该队列打开的进程也可以使用。