Microsoft Microsoft Microsoft

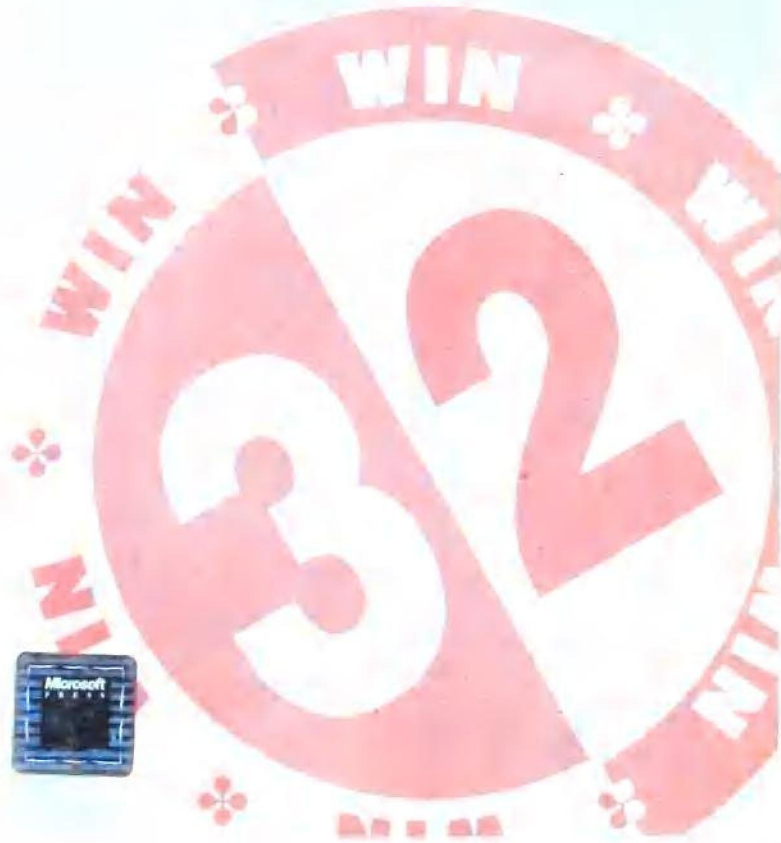
Microsoft® Win 32™ 程序员参考大全（三）

—— 函数(A—G)

[美] Microsoft Corporation 著

李 澈 周 灵 王慧敏 程艳红 刘德云 译

易 佳 审校



清华大学出版社

Microsoft® Win32™程序员参考大全(三)

——函 数 (A—G)

[美] Microsoft Corporation 著
李 澈 周 灵 王慧敏
程艳红 刘德云 译
易 佳 审校

清华大学出版社

(京)新登字 158 号

Microsoft® Win32™程序员参考大全(三)
——函数(A—G)

Microsoft® Win32™ Programmer's Reference, Volume 3,
——Functions A—G

Microsoft Corporation

图书在版编目(CIP)数据

Microsoft Win32 程序员参考大全(三)——函数(A—G)/美国微软公司(Microsoft Co.)著;李澈等译 .—北京:清华大学出版社,1995.4
ISBN 7-302-01675-5

Ⅰ.M… Ⅱ.①美… ②李… Ⅲ.软件开发-手册 Ⅳ.TP 311.52

中国版本图书馆 CIP 数据核字(95)第 15101 号

出版者:清华大学出版社(北京清华大学校内,邮编 100084)
印刷者:清华大学印刷厂
发行者:新华书店总店北京科技发行所
开 本:787×1092 1/16 印张:40 字数:947 千字
版 次:1995 年 5 月第 1 版 1995 年 5 月第 1 次印刷
书 号:ISBN 7-302-01675-5/TP·721
印 数:0001—4000
定 价:72.00 元

关于本书的说明

引言

Microsoft Win32 应用程序编程接口(API)可使得应用程序充分利用 32 位 Microsoft Windows 操作系统的能力。遵照 Win32 API 所写的应用程序可跨越单处理器系统和多处理器系统,并可移植到 RISC 体系结构上。本套书共分五卷,全面地剖析了 Microsoft Win32 API,整个文档包括窗口管理、图形、文件输入/输出、线程、内存管理、安全性、网络特性以及函数、消息、结构和宏列表。

本卷按字母顺序(H 到 Z)描述了 Win32 函数,定义了每一函数的语法、目的、参数和可能的返回值。许多函数的描述中还包括了一些附加的信息,例如函数是否支持 Unicode 字符集,以及函数是新的、已废弃的、还是已被删除的。

Microsoft Windows 和 C 编程语言

对于基于 Windows 的应用程序,C 编程语言是优选的开发语言。基于 Windows 的应用程序也可以用其它语言来开发,但 C 语言是可用来访问 Windows 函数的最直接、最简单的语言。因此,本书的所有语法和程序范例都是用 C 编程语言来写的。

Win32 API 使用了许多不属于标准 C 语言的类型、宏和结构。定义这些类型、宏和结构是为了更容易地创建基于 Windows 的应用程序,也为了使应用程序源更清楚和更易于理解。本书中讨论的所有类型、宏和结构都定义在 Win32 C 语言的头文件中。

文本约定

下列约定用于本手册的语法定义

约 定	含 义
斜体字符	指出某一位置或变量;其实际值需要提供。例如,SetCursorPos(*x*, *y*)语句就需要用具体的值来代替参数 *x* 和 *y*。
[ ]	其中是任选参数。
|	表示可以选择该竖杠所分隔开的两项中的一项。
…	说明前面的项可以多次重复。
⋮	表示示例应用程序中的省略部分。

Win32 应用程序编程接口(API)中的函数(A—G)

AbnormalTermination

New

BOOL AbnormalTermination (VOID)

该函数指明 try-finally 语句体的 try 块是否正常终止。只可在 try-finally 语句体的 finally 块中调用此函数。

参　数　本函数无参数。

返回值　如果由于发生异常而导致 try-finally 语句体的 try 块被终止,则返回值为 TRUE;否则,返回值为 FALSE。

注　释　只有在当执行完块中最后一条语句后执行体顺序离开块的情况下,try 块才会正常终止。会引起执行体离开 try 块的语句(诸如 return, goto, continue 或 break)将导致块的异常终止。即使这样的语句是 try 块中的最后一条语句,这种情况也会出现。

try 块的异常终止会引起系统向后搜索所有栈框(stack frame)以确定是否必须调用终止句柄。这将导致成百条指令的执行,所以避免由于 return, goto, continue 或 break 语句而引起的 try 块的异常终止是很重要的。注意,即使 try 块异常终止,这些语句也不会出现异常。

AbortDoc

int AbortDoc (*hdc*)

HDC *hdc*; / * handle of device context * /

该函数终止当前打印作业并删除自最后一次调用 StartDoc 函数以来写到设备上的所有信息。该函数取代了 ABORTDOC 打印机转义(escape)。

参　数　*hdc*

标识打印作业的设备描述表。

返回值　如果函数成功,则返回值大于 0;否则,返回值为 SP_ERROR。调用 GetLastError 函数,可得到更详细的错误信息。

注　释　应用程序应调用 AbortDoc 函数以在发生错误时,或在用户取消该作业后,终止一打印作业。要结束一个成功的打印作业,应用程序应该调用 EndDoc 函数。

如果用 Windows Print Manager(打印管理器)来启动打印作业,那么调用 Abort-

Doc 函数可擦除整个假脱机作业，这样打机机就接收不到任何内容。如果不是用打印管理程序来启动打印作业，那么数据有可能已被发送到打印机上了。在这种情况下，打印机驱动程序重新设置打印机(如果可能的话)，并结束打印作业。

参 见 **EndDoc, SetAbortProc , StartDoc**

AbortPath

New

BOOL AbortPath (*hdc*)

HDC *hdc*; / * handle of device context * /

该函数关闭并抛弃给定设备描述表(DC)中的所有路径。

参 数 *hdc*

标识将从中抛弃一路径的设备描述表(DC)。

返回值 如果函数成功，返回值为 TRUE；否则为 FALSE。要得到更详细的错误信息，可使用 GetLastError 函数。

注 释 如果在给定的设备描述表中有一打开的路径括号(bracket)，则该路径括号被关闭，该路径被抛弃。如果设备描述表中有一关闭的路径，则该路径被抛弃。

参 见 **BeginPath, EndPath**

AbortPrinter

New

BOOL AbortPrinter(*hPrinter*)

HANDLE *hPrinter*; / * handle of printer object * /

如果打印机被配置成用于假脱机，那么该函数可用来删除打印机的假脱机文件。

参 数 *hPrinter*

标识从中删除假脱机文件的打印机。

返回值 如果函数成功，则返回值为 TRUE；否则为 FALSE。调用 GetLastError 函数可得到更详细的错误信息。

注 释 如果打印机未被配置成用于假脱机，则该函数不起作用。

hPrinter 句柄由 OpenPrinter 函数返回。

参 见 **OpenPrinter**

AbortProc

BOOL CALLBACK AbortProc(*hdc*, *iError*)

HDC *hdc*; /* handle of device context */

int *iError*; /* error value */

该函数是应用程序定义的回调函数,当在假脱机期间要取消一打印作业时,可调用该函数。

参 数 *hdc*

标识设备描述表。

iError

指定是否发生了错误。如果没发生错误,则该参数为 0;如果 Windows Print-Manager 当前在磁盘空间外,且如果应用程序等待的话,会有更多可用的磁盘空间,则该参数为 SP_OUTOFDISK。

返回值 该回调函数应该返回,TRUE 以继续打印作业;或返回 FALSE 以取消打印作业。

注 释 应用程序通过调用 SetAbortProc 函数来安装此回调函数。AborProc 是该应用程序定义的函数名的占位符。

如果参数 *iError* 为 SP_OUTOFDISK,则应用程序不必取消打印作业。如果它不取消打印作业,则它必须通过调用 PeekMessage 或 GetMessage 函数来产生打印管理程序。

参 见 **GetMessage, PeekMessage, SetAbortProc**

AbortSystemShutdown

New. Unicode

BOOL AbortSystemShutdown(*lpszMachineName*)

LPTSTR *lpszmachineName*; / * addr. of name of computer to stop shutting down */

该函数终止一通过使用 InitiateSystemShutdown 函数来启动的系统的停机。

参 数 *lpszMachineName*

指向一个以 NULL 结束的字符串,该字符串指定将在其上终止停机的计算机的网络名字。如果 *lpszMachineName* 为 NULL 或指向一空串,则 AbortSystemShutdown 函数终止本地计算机上的停机。

返回值 如果函数成功,则返回值为 TRUE;否则,为 FALSE。通过调用 GetLastError 函数可得到更详细的错误信息。

注 释 InitiateSystemShutdown 函数显示一对话框告知用户系统正在停机。在 InitiateSystemShutdown 函数超时期间,AbortSystemShutdown 函数能够防止系统

停机。

要终止本地计算机停机，调用过程必须具有 SE_SHUTDOWN_NAME 特权。要终止远程计算机停机，调用过程必须对该远程计算机具有 SE_REMOTE_SHUTDOWN_NAME 特权。缺省情况下，用户具有对他们所登录的计算机的 SE_SHUTDOWN_NAME 特权，管理者具有对远程计算机的 SE_REMOTE_SHUTDOWN_NAME 特权。

该函数的失败一般是由于无效的计算机名字、不可访问的计算机或不充分的特权引起的。

参 见 **InitiateSystemShutdown**

AccessCheck

New

BOOL AccessCheck(*pSD*, *hClientToken*, *dwDesiredAccess*, *pgm*, *pps*, *cbps*, *lpdwGrantedAccess*, *lpfStatus*)

```
PSECURITY_DESCRIPTOR pSD;          /* address of security descriptor          */
HANDLE hClientToken;               /* handle of client access token           */
DWORD dwDesiredAccess;             /* access mask to request                  */
PGENERIC_MAPPING pgm;              /* address of generic-mapping structure    */
PPRIVILEGE_SET pps;                /* address of privilege-set structure      */
LPDWORD cbps;                      /* size of privilege-set structure         */
LPDWORD lpdwGrantedAccess;         /* address of granted access mask          */
LPBOOL lpfStatus;                  /* address of flag indicating whether access
                                      granted                                 */
```

该函数被服务器应用程序用来根据与对象相关的访问控制验证客户对该对象的访问权限。

参 数 *pSD*

指向 SECURITY_DESCRIPTOR 结构的指针，根据这个指针实施访问检查。

hClientToken

标识一个表示客户试图获得访问权的访问标记。这个句柄必须从通信对话层——比如，一个有名的管理——得到，以防止有可能发生的违反安全策略的行为。

dwDesiredAccess

指定所需的访问掩码。它必须先前已被 MapGenericMask 函数映射，以不包含任何类属访问权限。

pgm

指向与要检查的对象类型相关的 GENERIC_MAPPING 结构的指针。

GENERIC_MAPPING 结构有如下形式：

```
typedef struct _GENERIC_MAPPING { /* gm */
    ACCESS_MASK GenericRead;
    ACCESS_MASK GenericWrite;
    ACCESS_MASK GenericExecute;
    ACCESS_MASK GenericAll;
} GENERIC_MAPPING;
```

有关该结构的完整描述，参见《Microsoft Win32 程序员参考大全(五)》。

pps

指向一 PRIVILEGE_SET 结构，该函数将用来进行访问验证的特权填入此结构中。如果没有使用任何特权，则缓冲区中包含由零个特权组成的特权设置。

PRIVILEGE_SET 结构有如下形式：

```
typedef struct _PRIVILEGE_SET { /* ps */
    DWORD PrivilegeCount;
    DWORD Control;
    LUID_AND_ATTRIBUTES Privilege[ANYSIZE_ARRAY];
} PRIVILEGE_SET;
```

有关此结构的完整描述，参见《Microsoft Win32 程序员参考大全(五)》。

cbps

以字节为单位指定由参数 *pps* 指向的缓冲区的大小。

lpdwGrantedAccess

指向该函数用访问掩码来填充的变量，访问掩码指明所授予的访问权限。如果函数失败，则不提供访问掩码。

lpfStatus

指向一个指明访问检查是成功还是失败的标志。如果 *lpfStatus* 是 TRUE，则访问标记具有对该对象所要求的访问权限；如果 *lpfStatus* 失败，则该访问标记不具有所需的访问权限。当该参数是 FALSE 时，应用程序可使用 GetLastError 函数获得更详细的错误信息。

返回值 如果函数成功，则返回值为 TRUE；否则为 FALSE。通过 GetLastError 函数可得到更详细的错误信息。

注　释 该函数将指定的安全描述符与指定的访问标记进行比较，并通过 *lpfStatus* 参数指出访问是被允许还是被拒绝。如果允许访问，则所有要求的访问掩码就成为该对象的被允许的访问掩码。

在访问检查期间，只检查随意访问控制表(ACL)。

参　见 **AccessCheckAndAuditAlarm, AreAllAccessesGrantee AreAngAccessesGranted, MapGenericMask, PrivilegeCheck**

AccessCheckAndAuditAlarm New. Unicode

BOOL AccessCheckAndAuditAlarm(*lpszSubsystem*, *lpvHandleId*, *lpszObjectType*, *lpszObject*, *psd*, *dwDesiredAccess*, *pgm*, *fObjectCreation*, *lpdwGrantedAccess*, *lpfAccessStatus*, *lpfGenerateOnClose*)

```
LPTSTR lpszSubsystem;              /* address of string for subsystem name      */
LPVOID lpvHandleId;                /* address of handle identifier              */
LPTSTR lpszObjectType;             /* address of string for object type         */
LPTSTR lpszObject;                 /* address of string for object name         */
PSECURITY_DESCRIPTOR psd;          /* address of security descriptor            */
DWORD dwDesiredAccess;             /* mask for requested access rights          */
PGENERIC_MAPPING pgm;              /* address of GENERIC_MAPPING                */
BOOL fObjectCreation;              /* object-creation flag                      */
LPDWORD lpdwGrantedAccess;         /* address of mask for granted rights        */
LPBOOL lpfAccessStatus;            /* address of flag for results               */
LPBOOL lpfGenerateOnClose;         /* address of flag for audit generation      */
```

该函数执行一个访问验证并产生相应的查核消息,应用程序还可用此函数确定一客户过程是否有必要的特权。该函数通常被模仿一客户过程的服务器应用程序所使用。当前的 Windows NT 版本不支持报警(功能)。

参 数 *lpszSubsystem*

指向一以 NULL 结束的字符串,该串指定调用该函数的子系统的名字——例如,"DEBUG "或"Win32"。

lpvHandleID

指向一个与对象的客户句柄相应的唯一的 32 位值。如果访问被拒绝,则该值被忽略并可能被重新使用。

lpszObjectType

指向以 NULL 结束的字符串,该字符串指定要创建或访问的对象的类型。它出现在对象的查核日志中。

lpszObject

指向以 NULL 结束的字符串。该字符串指定要创建或访问的对象的名字。它出现在对象的查核日志中。

psd

指向 SECURITY_DESCRIPTOR 结构的指针,根据该指针实施访问检查。

dwDesiredAccess

指定给出所需的访问权限的访问掩码。该掩码必须先前已由 MapGeneric-Mask 函数映射,以不包含任何类属访问权限。

pgm

指向与要检查的对象的类型相关的 GENERIC_MAPPING 结构的指针。GENERIC_MAPPING 结构有如下形式：

```
tpyedef struct _GENERIC_MAPPING { /* gm */
    ACCESS_MASK GenericRead;
    ACCESS_MASK GenericWrite;
    ACCESS_MASK GenericExecute;
    ACCESS_MASK GenericAll;
} GENERIC_MAPPING;
```

有关该结构的完整描述，可参见《Microsoft Win32 程序员参考大全(五)》。

fObjectCreation

指定一个确定当允许访问时该调用应用程序是否将创建一个新对象的标志。如果此标志是 TRUE，则应用程序将创建一个新对象；如果为 FALSE，则应用程序打开一个已存在的对象。

lpdwGrantedAccess

指向一个缓冲区，当函数成功时，该缓冲区接收一个指明所授予的权限的访问掩码。

lpfAccessStatus

指向该函数设置的表明一个访问检查是成功还是失败的标志。如果允许访问，则此标志为 TRUE；否则为 FALSE。

lpfGenerateOnClose

指向一个当函数返回时由审核生成例程设置的标志。当关闭对象句柄时，必须把这个布尔值传递到 ObjectCloseAuditAlarm 上。

返回值 如果函数成功，则返回值为 TRUE；否则为 FALSE。通过 GetLastError 函数可得到更详细的错误信息。

注　释 该函数将指定的安全描述符与调用进程的模仿访问标记进行比较，并指出访问是否被允许或拒绝。如果允许访问，则所要求的访问掩码就成为该对象的允许访问掩码。作为试图访问的结果，该函数也将产生任何必要的审核消息。

这个函数需要调用进程有 SE_AUDIT_NAME 特权。对于该特权的测试总是对应于调用进程的原始标记，而不是该线程的私有标记进行的。

参　见 **AccessCheck, AreAllAccessesGranted, AreAnyAccessesGranted, MapGenericMask, ObjectCloseAuditAlarm, ObjectOpenAuditAlarm, ObjectPrivilegeAuditAlarm, PrivilegeCheck, PrivilegedServiceAuditAlarm**

AccessResource

该函数已经被删除，对于 Win32 应用程序，可使用 LoadResource, LockResource

和 FindResource 函数。

ActivateKeyboardLayout

New

BOOL ActivateKeyboardLayout(*hkl*, *fuFlags*)
HKL *hkl*; /* handle of keyboard layout */
UINT *fuFlags*; /* keyboard layout flags */

该函数激活一个不同的键盘布局。

参 数 *hkl*

标识要被激活的键盘布局。该布局必须已由先前对 LoadKeyboardLayout 函数的调用装载。此参数必须是一键盘布局的句柄或以下值之一：

值	意 义
HKL_NEXT	在由系统维护的已装载的布局的循环列表中选择下一个布局。
HKL_PREVIOUS	在由系统维护的已装载的布局的循环列表中选择前一个布局。

fuFlags

指定键盘布局是如何被激活的。此参数可以是以下值之一：

值	意 义
KLF_REORDER	如果设置该位，则系统的已装载键盘布局的循环列表将被重新排序。如果未设置该位，则循环列表中的次序不改变。例如，如果某用户有一个活动的英语布局，还有已装载的法语、德语和西班牙语布局(按此顺序)，则设置 KLF_REORDER 位来激活德语布局会产生以下顺序：德语、英语、法语、西班牙语。不设置 KLF_REORDER 位来激活德语布局会产生以下顺序：德语、西班牙语、英语、法语。
KLF_UNLOADPREVIOUS	以前激活的布局被卸载。

如果装载的布局少于三个，则此标志的值是无关的。

返回值 如果函数成功，则返回值为 TRUE；否则为 FALSE。调用 GetLastError 函数可得到更详细的错误信息。

注 释 可同时装载几个键盘布局，但一次只能激活一个。装载多个键盘布局使在布局间进行快速切换成为可能。

参 见 **LoadKeyboardLayout**

AddAccessAllowedAce

New

BOOL AddAccessAllowedAce(*pAcl*, *dwAclRevision*, *dwAccessMask*, *pSid*)

PACL *pAcl*; /* address of access-control list */

DWORD *dwAclRevision*; /* ACL revision level */

DWORD *dwAccessMask*; /* access mask */

PSID *pSid* /* address of security identifier */

该函数把一个允许访问的 ACE 增加到 ACL 上。允许对一个指定的 SID 的访问。一个 ACE 是一个访问控制项。一个 ACL 是一张访问控制表。一个 SID 是一个安全标识符。

参 数 *pAcl*

指向一个 ACL 结构。该函数对此 ACL 增加一个允许访问的 ACE。此 ACE 具有 ACCESS_ALLOWED_ACE 结构的形式。

ACL 结构具有以下形式：

```
typedef struct _ACL { /* acl */
    BYTE AclRevision;
    BYTE Sbz1;
    WORD AclSize;
    WORD AceCount;
    WORD Sbz2;
} ACL;
```

ACCESS_ALLOWED_ACE 结构具有以下形式：

```
typedef struct _ACCESS_ALLOWED_ACE { /* aaace */
    ACE_HEADER Header;
    ACCESS_MASK Mask;
    DWORD SidStart;
} ACCESS_ALLOWED_ACE;
```

有关这些结构的完整描述，参见《Microsoft Win32 程序员参考大全(五)》。

dwAclRevision

指定被修改的 ACL 的修正级。当前，该值必须是 ACL_REVISION。

dwAccessMask

指定授予指定的 SID 的访问权限的掩码。

pSid

指向 SID 结构，该结构代表被允许访问的进程。

返回值 如果函数成功，则返回值为 TRUE；否则为 FALSE。调用 GetLastError 函数可得到更详细的错误信息。

注 释 把一个允许访问的 ACE 增加到一个 ACL 上是修改 ACL 的最普通的方式。

被此函数放进 ACE 中的 ACE_HEADER 结构指定一个类型和大小，但不提供任何继承值和 ACE 标志。ACE_HEADER 结构具有以下形式：

```
typedef struct _ACE_HEADER { /* acehdr */
    BYTE AceType;
    BYTE AceFlags;
    WORD AceSize;
} ACE_HEADER;
```

有关此结构的完整描述，参见《Microsoft Win32 程序员参考大全(五)》。

参　见　**AddAccessDeniedAce, AddAce, AddAuditAccessAce, DeleteAce, GetAce**

AddAccessDeniedAce

New

BOOL AddAccessDeniedAce(*pAcl*, *dwAclRevision*, *dwAccessMask*, *pSid*)

PACL *pAcl*; /* address of access-control list */

DWORD *dwAclRevision*; /* ACL revision level */

DWORD *dwAccessMask*; /* access mask */

PSID *pSid*; /* address of security identifier */

本函数向一个 ACL 中添加一个拒绝访问的 ACE。该访问被一个指定的 SID 拒绝。ACE 是一个访问控制入口。ACL 是一个访问控制列表。SID 是一个安全标识符。

参　数　*pAcl*

指向一个 ACL 结构。本函数向该 ACL 中添加一个拒绝访问的 ACE。ACE 具有 ACCESS_DENIED_ACE 结构的形式。

ACL 结构有下面的形式：

```
typedef struct _ACL { /* acl */
    BYTE AclRevision;
    BYTE Sbz1;
    WORD AclSize;
    WORD AceCount;
    WORD Sbz2;
} ACL;
```

ACCESS_DENIED_ACE 结构具有下面的形式：

```
typedef struct _ACCESS_DENIED_ACE { /* adace */
    ACE_HEADER  Header;
    ACCESS_MASK Mask;
    DWORD       SidStart;
} ACCESS_DENIED_ACE;
```

有关这些结构的完整描述，参见《Micrososft Win32 程序员参考大全(五)》。

dwAclRevision

指定将被修改的 ACE 的修正级别。当前,该值必须是 ACL_REVISION。

dwAccessMask

指定访问权限的掩码,该访问权限被所指定的 SID 拒绝。

pSid

指向 SID 结构,该结构表示被拒绝访问的进程。

返回值 如果函数成功,返回值为 TRUE;否则,返回值为 FALSE。调用 GetLastError 函数,可获得进一步的错误信息。

注 释 被该函数放置在 ACE 中的 ACE_HEAOER 结构指定了一个类型和大小,但不提供任何 ACE 标志。ACE_HEADER 结构具有下面的形式:

```
typedef struct _ACE_HEADER { /* acehdr */
    BYTE AceType;
    BYTE AceFlags;
    WORD AceSize;
} ACE_HEADER;
```

有关该结构的完整描述,参见《Microsoft Win32 程序员参考大全(五)》。

参 见 **AddAccessAllowedAce, AddAce, AddAuditAccessAce, DeleteAce, GetAce**

AddAce

New

BOOL AddAce(*pAcl*, *dwAclRevision*, *dwStartingAceIndex*, *lpvAceList*, *cbAceList*)

```
PACL pAcl;                  /* address of access-control list      */
DWORD dwAclRevision;        /* ACL revision level                  */
DWORD dwStartingAceIndex;   /* index of ACE position in ACL        */
LPVOID lpvAceList;          /* address of one or more ACEs         */
DWORD cbAceList;            /* size of buffer for ACEs;            */
```

该函数向一个指定的 ACL 中增加一个或多个 ACE。

ACE 是一个访问控制入口。ACL 是一个访问控制列表。

参 数 *pAcl*

指向一个 ACL 结构。本函数向该 ACL 中增加一个 ACE。ACL 结构具有下面的形式:

```
typedef struct _ACL { /* acl */
    BYTE AclRevision;
    BYTE Sbz1;
    WORD AclSize;
    WORD AceCount;
    WORD Sbz2;
} ACL;
```

有关该结构的完整描述，参见《Microsoft Win32 程序员参考大全(五)》。

dwAclRevision

指定将被修改的 ACE 的修正级别。当前，该值必须是 ACL_REVISION。

dwStartingAceIndex

指定在 ACE 的 ACL 列表中添加新 ACE 的位置。该参数为零表示在列表开头插入 ACE。该参数值为 MAXDWORD 表示在列表末尾添加 ACE。

lpvAceList

指向一个包含一个或多个将被添加到指定的 ACL 中的 ACE 的列表。列表中的 ACE 必须被连续存储。

cbAceList

以字节为单位指定由 lpvAceList 参数所指向的输入缓冲区的大小。

返回值 如果函数成功，返回值为 TRUE；否则，返回值为 FALSE。调用 GetLastError 函数，可获得进一步的错误信息。

注　释 当使用 AddAce 函数来操作一个 ACL 时，应用程序通常使用 FindFirstFreeAce 和 GetAce 函数。此外，通过调用 GetAclInformation 函数所获取的 ACL_SIZE_INFORMATION 结构包含此 ACL 的大小和它们所包含的 ACE 的数目。

参　见 **AddAccessAllowedAce, AddAccessDeniedAce, AddAuditAccessAce, DeleteAce, FindFirstFreeAce, GetAce, GetAclInformation**

AddAtom

Unicode

ATOM AddAtom(*lpszName*)

LPCTSTR *lpszName*;　　/* address of string to add */

该函数向本地原子表增加一个字符串，并返回一个标识该字符串的唯一值(一个原子)。

参　数 *lpszName*

指向一个以 NULL 结束的被添加的字符串。只有大小写不同的字符串被认为是相同的。被添加的第一个字符串的大小写被保留，并由 GetAtomName 函数返回。

返回值 如果函数成功，返回值为 TRUE；否则，返回值为 FALSE。调用 GetLastError 函数，可获得进一步的错误信息。

注　释 该函数在原子表中存储一个给定的字符串的不超过一个的拷贝。如果该字符串已存在于此表中，则本函数返回该现有的原子；如果一个字符串原子已存在于该表中，则本函数增加此字符串的引用计数。

MAEKINTATOM 宏可被用来将一个 WORD 值转换成一个字符串，该字符串可被用 AddAtom 函数添加进该原子表中。

AddAtom 函数返回一个字符串原子，此字符串原子的值在 0xC000 和 0xFFFF

范围内。

如果 *lpszName* 具有"#1234"的形式，AddAtom 返回一个整数原子，它的值是在字符串中所指定的十进制数的 16 位的表示值(在此例中，是 0x04D2)。如果该所指定的十进制值是 0x0000 或者是 0xC000 到 0xFFFF 之间的值，则返回值是 0，表明出错。如果 *lpszName* 是 0x0001 到 0xBFFF 之间的值，则返回值是 *lpszName* 的低位字。

参 见 **DeleteAtom, FindAtom, GetAtomName, GlobalAddAtom, GlobalDeleteAtom, GlobalFindAtom, GlobalGetAtomName; MAKEINTATOM**

AddAuditAccessAce

New

BOOL AddAuditAccessAce (*pAcl*, *dwAclRevision*, *dwAccessMask*, *pSid*, *fAuditSuccess*, *fAuditFailure*)

```
PACL pAcl;                  /* address of access-control list                  */
DWORD dwAclRevision;        /* ACL revision level                              */
DWORD dwAccessMask;         /* access mask                                     */
PSID pSid;                  /* address of security identifier                  */
BOOL fAuditSuccess;         /* flag for auditing successful access             */
BOOL fAuditFailure;         /* flag for auditing unsuccessful access attempts */
```

该函数向一个系统 ACL 中添加一个系统查核 ACE。一个指定的 SID 的访问被查核。

ACE 是一个访问控制入口。ACL 是一个访问控制列表。SID 是一个安全标识符。

参 数 *pAcl*

指向一个 ACL 结构。本函数向该 ACL 中添加一个系统查核 ACE。ACE 具有 SYSTEM_AUDIT_ACE 结构的形式 ACL 结构具有下列形式：

```
typedef struct _ACL { /* acl */
    BYTE AclRevision
    BYTE Sbz1;
    WORD AclSize;
    WORD AceCount;
    WORD Sbz2;
} ACL;
```

SYSTEM_AVDIT_ACE 结构具有下列形式：

```
typedef struct _SYSTEM_AUDIT_ACE { /* sada */
    ACE_HEADER   Header;
    ACCESS_MASK  Mask;
    DWORD        SidStart;
```