BOOL AdjustTokenPrivileges(

HANDLE TokenHandle, // handle to token

BOOL DisableAllPrivileges, // disabling option

PTOKEN\_PRIVILEGES NewState, // privilege information

DWORD BufferLength, // size of buffer

PTOKEN\_PRIVILEGES PreviousState, // original state buffer

PDWORD ReturnLength // required buffer size

);

第一个参数是访问令牌的句柄；

第二个参数决定是进行权限修改还是丧失（Disable）所有权限；

第三个参数指明要修改的权限，是一个指向TOKEN\_PRIVILEGES结构的指针，该结构包含一个数组，数据组的每个项指明了权限的类型和要进行的操作;

typedef struct \_TOKEN\_PRIVILEGES {

DWORD PrivilegeCount; //指的数组元素的个数

LUID\_AND\_ATTRIBUTES Privileges[];

} TOKEN\_PRIVILEGES, \*PTOKEN\_PRIVILEGES;

LUID\_AND\_ATTRIBUTES这个结构的内容，声明如下：

typedef struct \_LUID\_AND\_ATTRIBUTES {

LUID Luid;

DWORD Attributes;

} LUID\_AND\_ATTRIBUTES, \*PLUID\_AND\_ATTRIBUTES

第一个参数就是指权限的类型，是一个LUID的值，LUID只要保证局部唯一，就是指在系统的每一次运行期间保证是唯一的就可以了。另外和GUID相同的一点，LUID也是一个64位的值，相信大家都看过GUID那一大串的值，要知道一个权限对应的LUID值是多少，这就要用到另外一个API函数LookupPrivilegevalue

第二个参数就指明了我们要进行的操作类型，有三个可选项： SE\_PRIVILEGE\_ENABLED、SE\_PRIVILEGE\_ENABLED\_BY\_DEFAULT、SE\_PRIVILEGE\_USED\_FOR\_ACCESS。要使能一个权限就指定Attributes为SE\_PRIVILEGE\_ENABLED。

第四个参数是结构PreviousState的长度，如果PreviousState为空，该参数应为NULL；

第五个参数也是一个指向TOKEN\_PRIVILEGES结构的指针，存放修改前的访问权限的信息，可空；

最后一个参数为实际PreviousState结构返回的大小。在使用这个函数前再看一下TOKEN\_PRIVILEGES这个结构