Portable Systems Group

Windows NT Shared Resource Specification

Author: *Gary D. Kimura*

Revision 1.5, May 15, 1990

1. Introduction

This specification describes the **Windows NT** routines that implement multiple-readers, single-writer access to a shared resource. Access is controlled via a shared resource variable and a set of routines to acquire the resource for shared access (also commonly known as read access) or to acquire the resource for exclusive access (also called write access).

A resource is logically in one of three states:

- Acquired for shared access
- Acquired for exclusive access
- Released (i.e., not acquired for shared or exclusive access)

Initially a resource is in the released state, and can be acquired for either shared or exclusive access by a user.

A resource that is acquired for shared access can be acquired by other users for shared access. The resource stays in the acquired for shared access state until all users that have acquired it have released the resource, and then it becomes released. Each resource, internally, maintains a count of the number of users granted shared access.

A resource that is acquired for exclusive access cannot be acquired by other users until the single user that has acquired the resource for exclusive access releases the resource. However, a thread can recursively acquire exclusive access to the same resource without blocking.

The routines described in this specification do not return to the caller until the resource has been acquired. or the user has the option of having the procedure return immediately to the caller if the resource cannot be acquired with blocking. The procedure's return value then indicates if the resource has been acquired.

To help avoid starvation of a user requesting exclusive access to a resource, the procedures do not allow additional users shared access to a resource if there is a user waiting for exclusive access to the resource.

Also, when a user releases exclusive access to a resource, priority is given to those waiting for exclusive access over those waiting for shared access.

The **APIs** that implement a shared resource are the following:

ExInitializeResource - Initialize a resource and set its state to released
ExAcquireResourceShared - Acquire shared access to a resource
ExAcquireResourceExclusive - Acquire exclusive access to a resource
ExReleaseResource - Release a resource (shared or exclusive)
ExConvertSharedToExclusive - Convert from shared to exclusive access
ExConvertExclusiveToShared - Convert from exclusive to shared access
ExDeleteResource - Deletes (i.e., uninitializes) a resource

2. Initializing a Resource Variable

A resource variable can be initialized and its state set to released with the **ExInitializeResource** procedure.

VOID
ExInitializeResource (
    IN PERESOURCE *Resource*
    );

Parameters:

Resource – A pointer to the resource variable being initialized

A resource variable cannot be used by the other procedures until it has been initialized.

3. Acquiring a Resource for Shared Access

A user can acquire shared access to a resource with the **ExAcquireResourceShared** procedure.

BOOLEAN
ExAcquireResourceShared (
    IN PERESOURCE *Resource*,
    IN BOOLEAN *Wait*
    );

Parameters:

Resource - A pointer to the resource variable to be acquired for shared access

Wait - Indicates if the call is allowed to block in order to acquire the resource. A value of TRUE indicates that the call is allowed to wait for the resource and FALSE indicates that control is to return immediately even if the resource cannot be acquired.

Return Value:

BOOLEAN - Returns TRUE if the access to the resource has been acquired and FALSE otherwise. If the Wait input parameter is TRUE then this function will always return a value of TRUE. If the Wait input parameter is FALSE then this function will return TRUE if the resource was acquired without blocking and FALSE if the resource cannot be acquired without blocking.

If the Wait parameter is TRUE, then this procedure does not return to the caller until the resource has been acquired for shared access. When the user acquires the resource, the count of the number of shared access users is incremented by one.

If the caller's thread has previously acquired exclusive access to the resource then the call to **ExAcquireResourceShared** will automatically succeed.

4. Acquiring a Resource for Exclusive Access

A user can acquire exclusive access to a resource with the **ExAcquireResourceExclusive** procedure.

BOOLEAN
ExAcquireResourceExclusive (
IN PERESOURCE *Resource*,
IN BOOLEAN *Wait*
);

Parameters:

Resource - A pointer to the resource variable to be acquired for exclusive access

Wait - Indicates if the call is allowed to block in order to acquire the resource. A value of TRUE indicates that the call is allowed to wait for the resource and FALSE indicates that control is to return immediately even if the resource cannot be acquired.

Return Value:

BOOLEAN - Returns TRUE if the access to the resource has been acquired and FALSE otherwise. If the Wait input parameter is TRUE then this function will always return a value of TRUE. If the Wait input parameter is FALSE then this function will return TRUE if the resource was acquired without blocking and FALSE if the resource cannot be acquired without blocking.

If the Wait parameter is TRUE, then this procedure does not return to the caller until the resource has been acquired for exclusive access.

If the caller's thread has previously acquired exclusive access to the resource then subsequent calls to **ExAcquireResourceExclusive** will automatically succeed.

5. Releasing a Resource

A user can release access to a resource with the **ExReleaseResource** procedure. This procedure is for releasing either exclusive access or shared access to a resource.

VOID
ExReleaseResource (
    IN PERESOURCE *Resource*
    );

Parameters:

Resource - A pointer to the resource variable to be released

If the resource is acquired for shared access, the number of users with shared access to the resource is decremented by one. If the count is now zero, the resource is released and next user waiting for exclusive access to the resource acquires it.

If the resource is acquired for exclusive access then the count of the number of times it has been recursively acquired is decremented by one. If the count is now zero, the resource is released and the next user waiting for access to the resource acquires it.

6. Changing from Shared Access to Exclusive Access

A user that has shared access to a resource can change to exclusive access with the **ExConvertSharedToExclusive** procedure.

VOID
ExConvertSharedToExclusive (
    IN PERESOURCE *Resource*
    );

Parameters:

Resource - A pointer to the resource variable to be acquired for exclusive access

This procedure does not return to the caller until the resource has been acquired for exclusive access. This procedure is similar in function to releasing a shared resource and then acquiring it for exclusive access; however, in the case where only one user has the resource acquired with shared access, the conversion to exclusive access with **ExConvertSharedToExclusive** is more efficient.

It is an error to try to covert a resource to exclusive access that is not currently acquired for either shared or exclusive access.

7. Changing from Shared Access to Exclusive Access

A user that has exclusive access to a resource can change to shared access with the **ExConvertExclusiveToShared** procedure.

VOID
ExConvertExclusiveToShared (
    IN PERESOURCE *Resource*
    );

Parameters:

Resource - A pointer to the resource variable to be converted to shared access

This procedure does not return to the caller until the resource has been acquired for shared access. This procedure is similar in function to releasing an exclusive resource and then acquiring it for shared access; however the user calling

ExConvertExclusiveToShared does not relinquish access to the resource as the two step operation does.

It is an error to try to convert a resource to shared access that is not currently acquired for exclusive access, or has been acquired recusively.

8. Deleting a Resource Variable

A resource variable can be deleted (i.e., uninitialized) with the **ExDeleteResource** procedure.

VOID
ExDeleteResource (
    IN PERESOURCE *Resource*
    );

Parameters:

Resource — A pointer to the resource variable being deleted

The programmer is responsible for ensuring that no one is actively using the resource. After calling this procedure the resource cannot be used again unless it is reinitialized.

Revision History:

Original Draft 1.0, June 23, 1989

Revision 1.1, June 26, 1989

1. Added explanation regarding who gets priority when releasing a resource
2. Changes **PEXRESOURCE** to **PERESOURCE**

Revision 1.2, June 27, 1989

1. Changed **ExChangeSharedToExclusive** to **ExConvertSharedToExclusive**.
2. Added **ExConvertExclusiveToShared**.

Revision 1.3, July 10, 1989

1. Added **ExDeleteResource**

Revision 1.4, March 21, 1990

1. Changed **ExAcquireResourceShared** and **ExAcquireResourceExclusive** to take an additional Wait input parameter and to return a BOOLEAN function value.

Revision 1.5, May 15, 1990

1. Added recusive exclusive resource acquisition.