Filtrer par titre
Fonction FindFirstChangeNotificationW
Fonction FindFirstFileA
Fonction FindFirstFileExA
Fonction FindFirstFileExW
Fonction FindFirstFileNameW
Fonction FindFirstFileW
Fonction FindFirstStreamW
Fonction FindFirstVolumeW
Fonction FindNextChangeNotification
FindNextFileA, fonction
FindNextFileNameW, fonction
FindNextFileW, fonction
FindNextStreamW, fonction
FindNextVolumeW, fonction
Fonction FindVolumeClose
GetCompressedFileSizeA, fonction
GetCompressedFileSizeW, fonction
GetDiskFreeSpaceA, fonction
GetDiskFreeSpaceExA, fonction
GetDiskFreeSpaceExW, fonction
GetDiskFreeSpaceW, fonction
GetDiskSpaceInformationA, fonction
GetDiskSpaceInformationW, fonction
Télécharger le PDF

Fonction FindFirstFileA (fileapi.h)

Article • 28/02/2024

Commentaires

Dans cet article

Recherche dans un répertoire un fichier ou un sous-répertoire dont le nom correspond à un nom spécifique (ou à un nom partiel si des caractères génériques sont utilisés).

Pour spécifier des attributs supplémentaires à utiliser dans une recherche, utilisez la fonction FindFirstFileEx .

Pour effectuer cette opération en tant qu'opération traitée, utilisez la fonction FindFirstFileTransacted .

Syntaxe

C++ Copier

```
HANDLE FindFirstFileA(
  [in]  LPCSTR             lpFileName,
  [out] LPWIN32_FIND_DATAA lpFindFileData
);
```

Paramètres

`[in] lpFileName`

Répertoire ou chemin d'accès et nom du fichier. Le nom de fichier peut inclure des caractères génériques, par exemple un astérisque (*) ou un point d'interrogation (?).

Ce paramètre ne doit pas être **NULL**, une chaîne non valide (par exemple, une chaîne vide ou une chaîne qui ne contient pas le caractère null de fin), ni se terminer par une barre oblique inverse de fin (\).

Si la chaîne se termine par un caractère générique, un point (.) ou un nom de répertoire, l'utilisateur doit disposer d'autorisations d'accès à la racine et à tous les sous-répertoires sur le chemin d'accès.

Par défaut, le nom est limité à MAX_PATH caractères. Pour étendre cette limite à 32 767 caractères larges, ajoutez « \\ ?\ » au chemin d'accès. Pour plus d'informations, consultez Nommage de fichiers, de chemins et d'espaces de noms.

> **Conseil**
>
> À compter de Windows 10, version 1607, vous pouvez choisir de supprimer la limitation MAX_PATH sans précédencer « \\ ?\ ». Pour plus d'informations, consultez la section « Limitation maximale de la longueur du chemin d'accès » de **Naming Files, Paths et Namespaces** .

`[out] lpFindFileData`

Pointeur vers la structure WIN32_FIND_DATA qui reçoit des informations sur un fichier ou un répertoire trouvé.

Valeur retournée

Si la fonction réussit, la valeur de retour est un handle de recherche utilisé dans un appel suivant à FindNextFile ou FindClose, et le paramètre *lpFindFileData* contient des informations sur le premier fichier ou répertoire trouvé.

Si la fonction échoue ou ne parvient pas à localiser les fichiers de la chaîne de recherche dans le paramètre *lpFileName* , la valeur de retour est **INVALID_HANDLE_VALUE** et le contenu de *lpFindFileData* est indéterminé. Pour obtenir des informations détaillées sur l'erreur, appelez la fonction GetLastError.

Si la fonction échoue, car aucun fichier correspondant n'est trouvé, la fonction GetLastError retourne **ERROR_FILE_NOT_FOUND**.

Remarques

La fonction **FindFirstFile** ouvre un handle de recherche et retourne des informations sur le premier fichier que le système de fichiers trouve avec un nom correspondant au modèle spécifié. Il peut s'agir ou non du premier fichier ou répertoire qui apparaît dans une application de liste de répertoires (par exemple, la commande dir) lorsqu'on lui donne le même modèle de chaîne de nom de fichier. Cela est dû au fait que **FindFirstFile** ne trie pas les résultats de la recherche. Pour plus d'informations, consultez FindNextFile.

La liste suivante identifie d'autres caractéristiques de recherche :

- La recherche est effectuée strictement sur le nom du fichier, et non sur des attributs tels qu'une date ou un type de fichier (pour d'autres options, consultez FindFirstFileEx).
- La recherche inclut les noms de fichiers longs et courts.
- Une tentative d'ouverture d'une recherche avec une barre oblique inverse de fin échoue toujours.
- La transmission d'une chaîne, **d'une valeur NULL** ou d'une chaîne vide non valide pour le paramètre *lpFileName* n'est pas une utilisation valide de cette fonction. Dans ce cas, les résultats ne sont pas définis.

> **Note** Dans de rares cas ou sur un système fortement chargé, les informations d'attribut de fichier sur les systèmes de fichiers NTFS peuvent ne pas être à jour au moment de l'appel de cette fonction. Pour être sûr d'obtenir les attributs de fichier du système de fichiers NTFS actuels, appelez la fonction **GetFileInformationByHandle** .

Une fois le handle de recherche établi, vous pouvez l'utiliser pour rechercher d'autres fichiers qui correspondent au même modèle à l'aide de la fonction FindNextFile .

Lorsque le handle de recherche n'est plus nécessaire, fermez-le en utilisant la fonction FindClose , et non CloseHandle.

Comme indiqué précédemment, vous ne pouvez pas utiliser une barre oblique inverse de fin (\) dans la chaîne d'entrée *lpFileName* pour **FindFirstFile**. Par conséquent, il peut ne pas être évident de rechercher des répertoires racines. Si vous souhaitez afficher des fichiers ou obtenir les attributs d'un répertoire racine, les options suivantes s'appliquent :

- Pour examiner les fichiers d'un répertoire racine, vous pouvez utiliser « C :* » et parcourir le répertoire à l'aide de FindNextFile.
- Pour obtenir les attributs d'un répertoire racine, utilisez la fonction GetFileAttributes .

> **Note** Le dépassement de la chaîne « \\ ?\ » n'autorise pas l'accès au répertoire racine.

Sur les partages réseau, vous pouvez utiliser un *lpFileName* sous la forme suivante : « \\Server\Share* ». Toutefois, vous ne pouvez pas utiliser un *lpFileName* qui pointe vers le partage lui-même ; par exemple, « \\Server\Share » n'est pas valide.

Pour examiner un répertoire qui n'est pas un répertoire racine, utilisez le chemin d'accès à ce répertoire, sans barre oblique inverse de fin. Par exemple, un argument « C :\Windows » retourne des informations sur le répertoire « C :\Windows », et non sur un répertoire ou un fichier dans « C :\Windows ». Pour examiner les fichiers et les répertoires dans « C :\Windows », utilisez un *lpFileName* de « C :\Windows* ».

N'oubliez pas qu'un autre thread ou processus peut créer ou supprimer un fichier portant ce nom entre le moment où vous interrogez le résultat et le moment où vous agissez sur les informations. S'il s'agit d'un problème potentiel pour votre application, une solution possible consiste à utiliser la fonction CreateFile avec **CREATE_NEW** (qui échoue si le fichier existe) ou **OPEN_EXISTING** (qui échoue si le fichier n'existe pas).

Si vous écrivez une application 32 bits pour répertorier tous les fichiers d'un répertoire et que l'application peut être exécutée sur un ordinateur 64 bits, vous devez appeler la fonction Wow64DisableWow64FsRedirection avant d'appeler **FindFirstFile** et d'appeler Wow64RevertWow64FsRedirection après le dernier appel à FindNextFile. Pour plus d'informations, consultez Redirecteur de système de fichiers.

Si le chemin pointe vers un lien symbolique, la mémoire tampon WIN32_FIND_DATA contient des informations sur le lien symbolique, et non sur la cible.

Dans Windows 8 et Windows Server 2012, cette fonction est prise en charge par les technologies suivantes.

Agrandir le tableau

Technologie	Prise en charge
Protocole Server Message Block (SMB) 3.0	Oui
Basculement transparent SMB 3.0 (TFO)	Oui
SMB 3.0 avec partages de fichiers avec montée en puissance parallèle (SO)	Oui
Système de fichiers du volume partagé de cluster (CsvFS)	Oui
Système de fichiers résilient (ReFS)	Oui

Exemples

L'exemple C++ suivant montre une utilisation minimale de **FindFirstFile**.

C++ Copier

```
#include <windows.h>
#include <tchar.h>
#include <stdio.h>

void _tmain(int argc, TCHAR *argv[])
{
   WIN32_FIND_DATA FindFileData;
   HANDLE hFind;

   if( argc != 2 )
   {
      _tprintf(TEXT("Usage: %s [target_file]\n"), argv[0]);
      return;
   }

   _tprintf (TEXT("Target file is %s\n"), argv[1]);
   hFind = FindFirstFile(argv[1], &FindFileData);
   if (hFind == INVALID_HANDLE_VALUE)
   {
      printf ("FindFirstFile failed (%d)\n", GetLastError());
      return;
   }
   else
   {
      _tprintf (TEXT("The first file found is %s\n"),
                FindFileData.cFileName);
      FindClose(hFind);
   }
}
```

Pour un autre exemple, consultez Liste des fichiers dans un répertoire.

> **Notes**
>
> L'en-tête fileapi.h définit FindFirstFile comme un alias qui sélectionne automatiquement la version ANSI ou Unicode de cette fonction en fonction de la définition de la constante de préprocesseur UNICODE. Le mélange de l'utilisation de l'alias neutre en encodage avec du code qui n'est pas neutre en encodage peut entraîner des incompatibilités qui entraînent des erreurs de compilation ou d'exécution. Pour plus d'informations, consultez **Conventions pour les prototypes de fonction**.

Configuration requise

Agrandir le tableau

Condition requise	Valeur
Client minimal pris en charge	Windows XP [applications de bureau | applications UWP]
Serveur minimal pris en charge	Windows Server 2003 [applications de bureau | applications UWP]
Plateforme cible	Windows
En-tête	fileapi.h (inclure Windows.h)
Bibliothèque	Kernel32.lib
DLL	Kernel32.dll

Voir aussi

Fonctions de gestion des fichiers

FindClose

FindFirstFileEx

FindFirstFileTransacted

FindNextFile

GetFileAttributes

SetFileAttributes

Liens symboliques

Utilisation des en-têtes Windows

WIN32_FIND_DATA

Commentaires

Cette page a-t-elle été utile ? Yes No

Indiquer des commentaires sur le produit | Obtenir de l'aide sur Microsoft Q&A