The CreateFile Function

Mario Tayah and Jim Fawcett

CSE775 – Distributed Objects

Spring 2007

Definition

- The **CreateFile** function creates or opens anything that can be used as a source of data including:
  - file
  - file stream
  - Directory
  - physical disk
  - Volume
  - console buffer
  - tape drive
  - communications resource
  - Mailslot
  - named pipe
- This function returns a handle that can be used to access an object which represents what you have asked to open.

CreateFile Signature

- HANDLE CreateFile(

  LPCTSTR lpFileName,
  // pointer to a string that specifies name of an object to create or open

  DWORD dwDesiredAccess,
  // access to the object, which can be read, write, or both

  DWORD dwShareMode,
  // can be read, write, both, or none

  LPSECURITY_ATTRIBUTES lpSecurityAttributes,
  // pointer to a SECURITY_ATTRIBUTES structure that determines whether
  // or not the returned handle can be inherited by child processes

  DWORD dwCreationDisposition,
  // [in] An action to take on files that exist and that do not exist

  DWORD dwFlagsAndAttributes,
  // file attributes and flags

  HANDLE hTemplateFile
  // A handle to a template file with GENERIC_READ access rights.
  // The template file supplies file attributes and extended attributes
  // for the file that is being created. This parameter can be NULL

);

Signature (cont.)

- Return Values :
  - If the function succeeds, the return value is an open handle to the specified file.
  - If the function succeeds, the return value is an open handle to a specified file. If a specified file exists before the function call and *dwCreationDisposition* is CREATE_ALWAYS or OPEN_ALWAYS, a call to **GetLastError** returns ERROR_ALREADY_EXISTS, even when the function call succeeds.
  - If a file does not exist before the call, **GetLastError** returns 0 (zero).
  - If the function fails, the return value is INVALID_HANDLE_VALUE.
  - Note : To get detailed error information, call **GetLastError**.
- Note : always Use the CloseHandle function to close an object handle returned by the CreateFile function after you are done with using the file.

Use CreateFile to access files

- The CreateFile function can create a new file or open an existing file. You must specify the file name, creation instructions, and other attributes.
- On the right, you see two code fragments; one that creates a file for reading and the other creates a file for writing.

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

HANDLE hFile;

hFile = CreateFile(TEXT("myfile.txt"),    // file to open
                   GENERIC_READ,          // open for reading
                   FILE_SHARE_READ,       // share for reading
                   NULL,                  // default security
                   OPEN_EXISTING,         // existing file only
                   FILE_ATTRIBUTE_NORMAL, // normal file
                   NULL);                 // no attr. template

if (hFile == INVALID_HANDLE_VALUE)
{
    printf("Could not open file (error %d)\n", GetLastError());
    return 0;
}
```

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

HANDLE hFile;

hFile = CreateFile(TEXT("myfile.txt"),     // file to create
                   GENERIC_WRITE,          // open for writing
                   0,                      // do not share
                   NULL,                   // default security
                   CREATE_ALWAYS,          // overwrite existing
                   FILE_ATTRIBUTE_NORMAL | // normal file
                   FILE_FLAG_OVERLAPPED,   // asynchronous I/O
                   NULL);                  // no attr. template

if (hFile == INVALID_HANDLE_VALUE)
{
    printf("Could not open file (error %d)\n", GetLastError());
    return 0;
}
```

What does the CreateFile function do?

- When creating a new file, the CreateFile function performs the following actions:
  - Clears the existing file attributes (CREATE_ALWAYS with an existing file only).
  - Combines file attributes and flags specified by dwFlagsAndAttributes with FILE_ATTRIBUTE_ARCHIVE.
  - Sets the file length to zero.
  - Copies the extended attributes supplied by the template file to the new file if the hTemplateFile parameter is specified.
  - Sets the SD specified by the lpSecurityDescriptor member of the SECURITY_ATTRIBUTES structure (except when using CREATE_ALWAYS on an existing file).

- When opening an existing file, CreateFile performs the following actions:
  - Combines the file flags (FILE_FLAG_*) specified by dwFlagsAndAttributes with existing file attributes. CreateFile ignores the file attributes (FILE_ATTRIBUTE_*) specified by dwFlagsAndAttributes.
  - Sets the file length according to the value of dwCreationDisposition.
  - Ignores the hTemplateFile parameter.
  - Ignores the lpSecurityDescriptor member of the SECURITY_ATTRIBUTES structure. The other structure members are used (for example, bInheritHandle indicates whether the file handle can be inherited).

Use CreateFile to open filestreams

- The code to the right illustrates how to create a filestream using the createfile function.

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

void main( )
{
   HANDLE hFile, hStream;
   DWORD dwRet;

   hFile   = CreateFile( "testfile",
                         GENERIC_WRITE,
                         FILE_SHARE_WRITE,
                         NULL,
                         OPEN_ALWAYS,
                         0,
                         NULL );
   if( hFile == INVALID_HANDLE_VALUE )
      printf( "Cannot open testfile\n" );
   else
       WriteFile( hFile, "This is testfile", 16, &dwRet, NULL );

   hStream = CreateFile( "testfile:stream",
                         GENERIC_WRITE,
                         FILE_SHARE_WRITE,
                         NULL,
                         OPEN_ALWAYS,
                         0,
                         NULL );
   if( hStream == INVALID_HANDLE_VALUE )
      printf( "Cannot open testfile:stream\n" );
   else
      WriteFile(hStream,
                "This is testfile:stream", 23, &dwRet, NULL);
}
```

Use CreateFile to open Directories

- An application cannot create a directory by using CreateFile.
- The application must call CreateDirectory or CreateDirectoryEx to create a directory.
- However an application can open a directory by using CreateFile.
- To open a directory by using CreateFile, use the FILE_FLAG_BACKUP_SEMANTICS flag.

Use CreateFile to open physical disk drive or volume

- You can use the CreateFile function to open a physical disk drive or a volume.
- The function returns a handle that can be used with the DeviceIoControl function. This enables you to access the disk partition table.
- However, it is potentially dangerous to use the create file function to open a phsical drive/volume, because an incorrect write to a disk could make its contents inaccessible.
- The following requirements must be met for such a call to succeed:
  - The caller must have administrative privileges.
  - The dwCreationDisposition parameter must have the OPEN_EXISTING flag.
  - When opening a volume or floppy disk, the dwShareMode parameter must have the FILE_SHARE_WRITE flag.
  - When opening a physical drive x, the lpFileName string should be the following form: \\.\PHYSICALDRIVE<x>. Hard disk numbers start at 0 (zero).
- To the right is an example of using the createfile function to open a physical drive.

```
BOOL GetDriveGeometry(DISK_GEOMETRY *pdg)
{
  HANDLE hDevice;               // handle to the drive to be examined
  BOOL bResult;                 // results flag
  DWORD junk;                   // discard results

  hDevice = CreateFile("\\\\.\\PhysicalDrive0",  // drive to open
                    0,                // no access to the drive
                    FILE_SHARE_READ | // share mode
                    FILE_SHARE_WRITE,
                    NULL,             // default security attributes
                    OPEN_EXISTING,    // disposition
                    0,                // file attributes
                    NULL);            // do not copy file attributes

  if (hDevice == INVALID_HANDLE_VALUE) // cannot open the drive
  {
    return (FALSE);
  }

  bResult = DeviceIoControl(hDevice,  // device to be queried
      IOCTL_DISK_GET_DRIVE_GEOMETRY,  // operation to perform
                           NULL, 0, // no input buffer
                          pdg, sizeof(*pdg),     // output buffer
                          &junk,                 // # bytes returned
                          (LPOVERLAPPED) NULL);  // synchronous I/O

  CloseHandle(hDevice);

  return (bResult);
}
```

Use CreateFile to access tape drives

- Tapes are mostly obsolete now, yet they are still used for some backup activities,
  below is a list of operations that walk you through handling tape access.
  - An application must use the CreateFile function to create a handle of a tape device. This handle is used in subsequent operations on the tape in the device.
  - Before an application writes to a tape, the tape must be formatted according to the needs of the application and the capabilities of the tape drive being used. The CreateTapePartition function reformats a tape, creating on it a given number of partitions of a specified size.
  - The PrepareTape function prepares a tape to be accessed or removed. This function can load, unload, lock, or unlock a tape. This function can also tension the tape by moving the tape to the end of the tape and back to the beginning.
  - To retrieve and set information about a tape and tape drive, an application uses the GetTapeParameters, SetTapeParameters, and GetTapeStatus functions.
  - GetTapeParameters retrieves information that describes a tape or a tape drive. The tape information includes the tape's type, density, and block size; the number of partitions on the tape; the amount of tape remaining; and so on. The tape drive information includes the drive's default block size, the maximum partition count, and the features that are supported.
  - SetTapeParameters either sets the tape block size or sets the tape drive flags that indicate whether the drive supports hardware error correction, data compression, data padding, or any combination of the three.
  - GetTapeStatus indicates whether the tape drive is ready to process tape commands.

Use CreateFile to access communication ports

- The CreateFile function can create a handle to a communications resource, such as the serial port COM1. For communications resources, the dwCreationDisposition parameter must be OPEN_EXISTING, and the template parameter must be NULL. Read, write, or read/write access can be specified, and the handle can be opened for overlapped I/O.
- To the right is a code fragment that illustrates the usage of createfile for accessing communication ports

```
int main(int argc, char *argv[])
{
   DCB dcb;
   HANDLE hCom;
   BOOL fSuccess;
   char *pcCommPort = "COM2";

   hCom = CreateFile( pcCommPort,
                      GENERIC_READ | GENERIC_WRITE,
                      0,    // must be opened with exclusive-access
                      NULL, // no security attributes
                      OPEN_EXISTING, // must use OPEN_EXISTING
                      0,    // not overlapped I/O
                      NULL  // hTemplate must be NULL for comm devices
                      );

   if (hCom == INVALID_HANDLE_VALUE)
   {
       // Handle the error.
       printf ("CreateFile failed with error %d.\n", GetLastError());
       return (1);
   }

   fSuccess = GetCommState(hCom, &dcb);

   if (!fSuccess)
   {
      // Handle the error.
      printf ("GetCommState failed with error %d.\n", GetLastError());
      return (2);
   }

   // Fill in DCB: 57,600 bps, 8 data bits, no parity, and 1 stop bit.

   dcb.BaudRate = CBR_57600;     // set the baud rate
   dcb.ByteSize = 8;             // data size, xmit, and rcv
   dcb.Parity = NOPARITY;        // no parity bit
   dcb.StopBits = ONESTOPBIT;    // one stop bit

   fSuccess = SetCommState(hCom, &dcb);

   if (!fSuccess)
   {
      // Handle the error.
      printf ("SetCommState failed with error %d.\n", GetLastError());
      return (3);
   }

   printf ("Serial port %s successfully reconfigured.\n", pcCommPort);
   return (0);
}
```

Use CreateFile to access Console

- The CreateFile function can create a handle to console input (CONIN$).
- If the process has an open handle to it as a result of inheritance or duplication, it can also create a handle to the active screen buffer (CONOUT$).
- The calling process must be attached to an inherited console or one allocated by the Alloc console function.
- The table on the next slide illustrates the parameters that you have to use in the CreatFile function to use it to access the Console.

Use CreateFile to access Console

Parameters	Value
lpFileName	Use the CONIN$ value to specify console input. Use the CONOUT$ value to specify console output. CONIN$ gets a handle to the console input buffer, even if the SetStdHandle function redirects the standard input handle. To get the standard input handle, use the GetStdHandle function. CONOUT$ gets a handle to the active screen buffer, even if **SetStdHandle** redirects the standard output handle. To get the standard output handle, use **GetStdHandle**.
dwDesiredAccess	GENERIC_READ | GENERIC_WRITE is preferred, but either one can limit access.
dwShareMode	When opening CONIN$, specify FILE_SHARE_READ. When opening CONOUT$, specify FILE_SHARE_WRITE. If the calling process inherits the console, or if a child process should be able to access the console, this parameter must be FILE_SHARE_READ | FILE_SHARE_WRITE.
lpSecurityAttributes	If you want the console to be inherited, the **bInheritHandle** member of the SECURITY_ATTRIBUTES structure must be TRUE.
dwCreationDisposition	You should specify OPEN_EXISTING when using **CreateFile** to open the console.
dwFlagsAndAttributes	Ignored.
hTemplateFile	Ignored.

Use CreateFile to access MailSlots

- A mailslot is a mechanism for one-way interprocess communications (IPC).
- Applications can store messages in a mailslot.
- The owner of the mailslot can retrieve messages that are stored there.
- These messages are typically sent over a network to either a specified computer or to all computers in a specified domain.
- A domain is a group of workstations and servers that share a group name.
- You can use the CreateFile function to open the client end of a mailslot.
- the function returns INVALID_HANDLE_VALUE if the mailslot client attempts to open a local mailslot before the mailslot server has created it with the CreateMailSlot function.

Use CreateFile to access Pipes

- A pipe is a section of shared memory that processes use for communication.
- The process that creates a pipe is the pipe server. A process that connects to a pipe is a pipe client.
- One process writes information to the pipe, then the other process reads the information from the pipe.
- You can use the CreateFile function to open the client end of a named pipe.
- The function uses any instance of the named pipe that is in the listening state.
- The opening process can duplicate the handle as many times as required, but after it is opened, the named pipe instance cannot be opened by another client.
- The access that is specified when a pipe is opened must be compatible with the access that is specified in the dwOpenMode parameter of the CreateNamedPipe function.

End of Presentation