# 方法一. 不能监控具体变化

首先介绍几个重要的api函数：

FindFirstChangeNotification( )；

FindNextChangeNotification()；

WaitForSingleObject()；

其中FindFirstChangeNotification（lpzpath，fwatchsubtree,fdwfilter）

lpzpath表示要监视的路径名，

fwatchsubtree判断是否查看子目录,

fdwfilter为要监视的事件，函数执行成功后返回一个句柄。

参数fdwfilter取值及其含义如下：

FILE\_NOTIFY\_CHANGE\_FILE\_NAME //查看指定目录下任何文件名的改变

FILE\_NOTIFY\_CHANGE\_DIR\_NAME //查看指定目录下任何目录名的改变

FILE\_NOTIFY\_CHANGE\_SIZE // 查看指定目录下文件大小的改变

FILE\_NOTIFY\_CHANGE\_ATTRIBUTES //查看指定目录下文件属性的改变

FindNextChangeNotification（hchange）

hchange为FindFirstChangenNotification返回的句柄，其作用是请求系统在下次

检测到相应改变时发出改变通知消息句柄。当函数成功返回后，应用程序可通过WaitForMultipleObjects或WaitfForSingleObject来等待发生改变的通知。WaitForSingleObject（hchange，dwmilliseconds）中hchange为FindFirstChangeNotification　返回的句柄，dwmilliseconds为等待时间值，指定等待完成需要的时间，单位为毫秒。该值为－1时表示时间无限长。最好在结束监视程序之前先用 FindCloseChangeNotification(hchange)来关闭句柄。

# 方法二． 使用函数ReadDirectoryChangesW

该函数的原型为：

BOOL WINAPI ReadDirectoryChangesW(

\_\_in HANDLE hDirectory,//指向监控目录的句柄，可以用CreatFile生成

\_\_in\_out LPVOID lpBuffer,//存储修改信息的首地址

\_\_in DWORD nBufferLength,//分配的存储修改信息的空间的长度

\_\_in BOOL bWatchSubtree,//TRUE则监控子目录，FALSE则不监控

\_\_in DWORD dwNotifyFilter,//通知条件

\_\_out LPDWORD lpBytesReturned,//该函数返回信息的字节数，也就是存到lpBuffer中的内容的字节数

\_\_in LPOVERLAPPED lpOverlapped,//一个指向OVERLAPPED结构的指针，他在同步调用时提供数据供使用，否则他就为NULL

\_\_in LPOVERLAPPED\_COMPLETION\_ROUTINE lpCompletionRoutine//当操作结束或被取消或者线程进入等待状态时的一个指向将被调用操作的指针

);

返回值：如果函数成功，返回值就是非0。对于同步调用，这意味着操作成功，对于异步调用，这意味着操作成功地排队。如果函数失败，返回值是0。如果操作目录或文件系统不支持这个操作，函数将返回ERROR\_INVALID\_FUNCTION，可以使用GetLastError()函数获取。

lpBuffer指向修改信息的结构体如下：

typedef struct \_FILE\_NOTIFY\_INFORMATION {

DWORD NextEntryOffset; // 下一修改的地址

DWORD Action; //具体修改动作

DWORD FileNameLength; // 文件名字的长度

WCHAR FileName[1]; // 文件名首地址

} FILE\_NOTIFY\_INFORMATION, \*PFILE\_NOTIFY\_INFORMATION;

微软官方解释如下：

**NextEntryOffset**

Number of bytes that must be skipped to get the next record. A value of zero indicates that this is the last record.

**Action**

Type of change that occurred. The following table shows possible values.

|  |  |
| --- | --- |
| **Value** | **Description** |
| FILE\_ACTION\_ADDED | The file was added to the directory. |
| FILE\_ACTION\_MODIFIED | The file was modified. This can be a change in the time stamp or attributes. |
| FILE\_ACTION\_REMOVED | The file was removed from the directory. |
| FILE\_ACTION\_RENAMED\_NEW\_NAME | The file was renamed, and this value is the new name. |
| FILE\_ACTION\_RENAMED\_OLD\_NAME | The file was renamed, and this value is the old name. |

**FileNameLength**

Length of the file name portion of the record, in bytes. The length does not include the terminating NULL character.

**FileName**

File name, relative to the directory handle. The file name is in Unicode format and is not null-terminated.

[Requirements](javascript:void(0))

|  |  |
| --- | --- |
| Header | winnt.h |
| Windows Embedded CE | Windows CE 1.01 and later |
| Windows Mobile | Windows Mobile Version 5.0 and later |

# 项目中遇到的问题和解决思路

1. 项目中只需监测单个文件的内容是否修改，修改后做相应处理。而ReadDirectoryChangesW函数是用来监测文件夹的（传文件句柄，监测失败）。本文采取措施为，在监测到变化后根据文件名确定是否是需要监测的文件，即对监测结果进行过滤，只处理需要关注的文化的变化。
2. 若需监测文件名称的变化，FILE\_NOTIFY\_INFORMATION结构中NextEntryOffset的值一般不是0，需要计算出下一个变化信息的结构地址，使用迭代的方法进行处理
3. 当保存某个文件时，ReadDirectoryChangesW会响应多次，得到的变化信息结构体个数有多个，具体与文件操作工具有关，所以要在最后一次监测到文件变化后执行响应操作，具体为什么ReadDirectoryChangesW会在一次保存时多次响应 FILE\_ACTION\_MODIFIED，还未查到相关资料。

# 监控示例程序代码

typedef struct

{

CString m\_strDir;

}MonitorThreadParam;

CString CStrW2CStrA(const CStringW &cstrSrcW)

{

int len = WideCharToMultiByte(CP\_ACP, 0, LPCWSTR(cstrSrcW), -1, NULL, 0, NULL, NULL);

char \*str = new char[len];

memset(str, 0, len);

WideCharToMultiByte(CP\_ACP, 0, LPCWSTR(cstrSrcW), -1, str, len, NULL, NULL);

CString cstrDestA = str;

delete[] str;

return cstrDestA;

}

DWORD WINAPI NotifyThreadPro(LPVOID param)

{

// 获取要监控的目录的句柄

MonitorThreadParam \*Threadparam = (MonitorThreadParam\*)param;

CString strDir = Threadparam->m\_strDir;

HANDLE hDir = CreateFile( strDir, FILE\_LIST\_DIRECTORY,

FILE\_SHARE\_READ |

FILE\_SHARE\_WRITE |

FILE\_SHARE\_DELETE, NULL,

OPEN\_EXISTING, FILE\_FLAG\_BACKUP\_SEMANTICS |

FILE\_FLAG\_OVERLAPPED, NULL);

if (hDir == INVALID\_HANDLE\_VALUE)

{

printf("hDir:INVALID\_HANDLE\_VALUE\r\n");

return 1;

}

char notifyMsg[1024] = {0};

wchar\_t wstrFileName[256]={0};

CString strFileName;

DWORD dRetBytes = 0;

FILE\_NOTIFY\_INFORMATION \*pnotify=(FILE\_NOTIFY\_INFORMATION \*)notifyMsg;

FILE\_NOTIFY\_INFORMATION \*tmp;

while (TRUE)

{

\_strset\_s(notifyMsg,0);

if(!ReadDirectoryChangesW(hDir, &notifyMsg, sizeof(notifyMsg),

TRUE, FILE\_NOTIFY\_CHANGE\_FILE\_NAME|

FILE\_NOTIFY\_CHANGE\_DIR\_NAME|

FILE\_NOTIFY\_CHANGE\_ATTRIBUTES|

FILE\_NOTIFY\_CHANGE\_SIZE|

FILE\_NOTIFY\_CHANGE\_LAST\_WRITE|

FILE\_NOTIFY\_CHANGE\_LAST\_ACCESS|

FILE\_NOTIFY\_CHANGE\_CREATION|

FILE\_NOTIFY\_CHANGE\_SECURITY,

&dRetBytes, NULL, NULL))

continue;

tmp = pnotify;

while(1)

{

StrCpyNW(wstrFileName,tmp->FileName,tmp->FileNameLength/2 + 1);

strFileName = CStrW2CStrA(wstrFileName);

strFileName = strDir + "\\" + strFileName;

switch(tmp->Action)

{

case FILE\_ACTION\_ADDED:

printf("Directory/File added （添加文件 %s）- \r\n",(LPCTSTR)strFileName);

break;

case FILE\_ACTION\_REMOVED:

printf("Directory/File removed （删除文件 %s）- \r\n",(LPCTSTR)strFileName);

break;

case FILE\_ACTION\_MODIFIED:

printf("Directory/File modified （修改文件内容 %s）- \r\n",(LPCTSTR)strFileName);

break;

case FILE\_ACTION\_RENAMED\_OLD\_NAME:

printf("Directory/File old name （修改文件名字 %s）- \r\n",(LPCTSTR)strFileName);

break;

case FILE\_ACTION\_RENAMED\_NEW\_NAME:

printf("Directory/File new name （修改文件名字 %s）- \r\n",(LPCTSTR)strFileName);

break;

default:

break;

}

if (tmp->NextEntryOffset > 0)

{

// 计算下一个变化信息结构体的地址

tmp = (FILE\_NOTIFY\_INFORMATION\*)((char\*)tmp + tmp->NextEntryOffset);

}

else

break;

}

}

return 0;

}

void TestMonitor(CString strDir)

{

MonitorThreadParam param;

param.m\_strDir = strDir;

HANDLE hThread = CreateThread(NULL,0,NotifyThreadPro,&param,0,NULL);

WaitForSingleObject(hThread,INFINITE);

CloseHandle(hThread);

printf("监控退出...");

}

void main()

{

TestMonitor("D:\\Temp");

}