File System Watcher

Gregory Adam

2015

Le minerai de fer peut croire qu'il est torturé sans raison dans la fournaise, mais lorsque la lame de l'acier le plus fin réfléchit à cette torture, elle en comprend la raison.

(T. LOBSANG RAMPA)

Contents

- Contents
  - Purpose
  - Possibilities
  - Choices
    - Implementation
    - Messaging
    - Detect changes API
    - Program flow

Where are we

- Contents
  - Purpose
  - Possibilities
  - Choices
    - Implementation
    - Messaging
    - Detect changes API
    - Program flow

Purpose

Detect changes in one directory or more directories

- in an event-driven way
- without blocking the main thread

Where are we

- Contents
  - Purpose
  - Possibilities
  - Choices
    - Implementation
    - Messaging
    - Detect changes API
    - Program flow

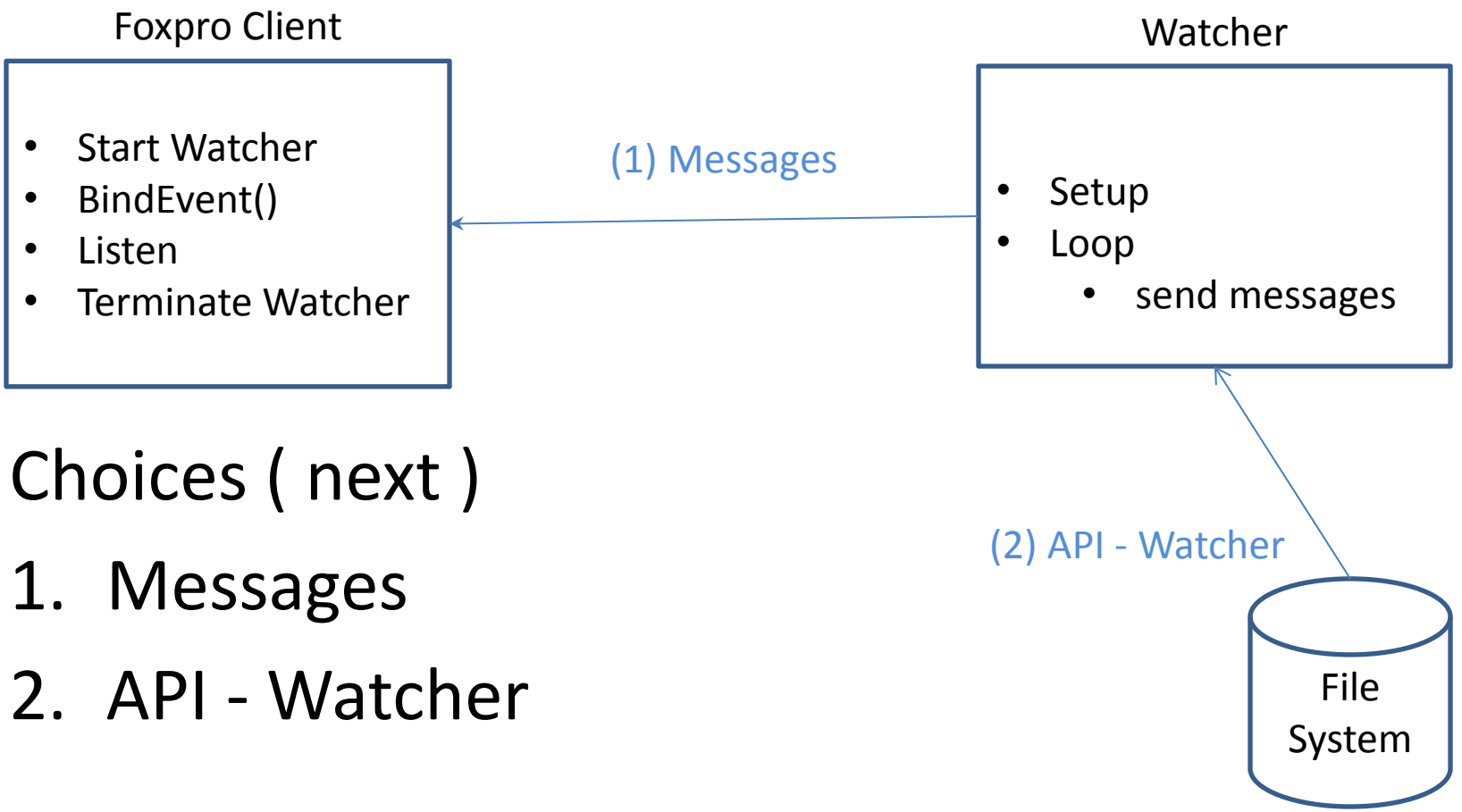
Possibilities

- FoxPro with a timer
- Separate process which sends messages since
  - we will need a loop (foxpro does not implement callbacks natively)

Possibilities - FoxPro with a timer

- Basically a loop
- Adir() + save
- Timer Event
  - Adir()
  - Compare old array to new – raise events for changes
  - Copy new array to old array
- Difficult to watch a tree of directories
- Hard to tell whether a file has been renamed

Possibilities – Separate process



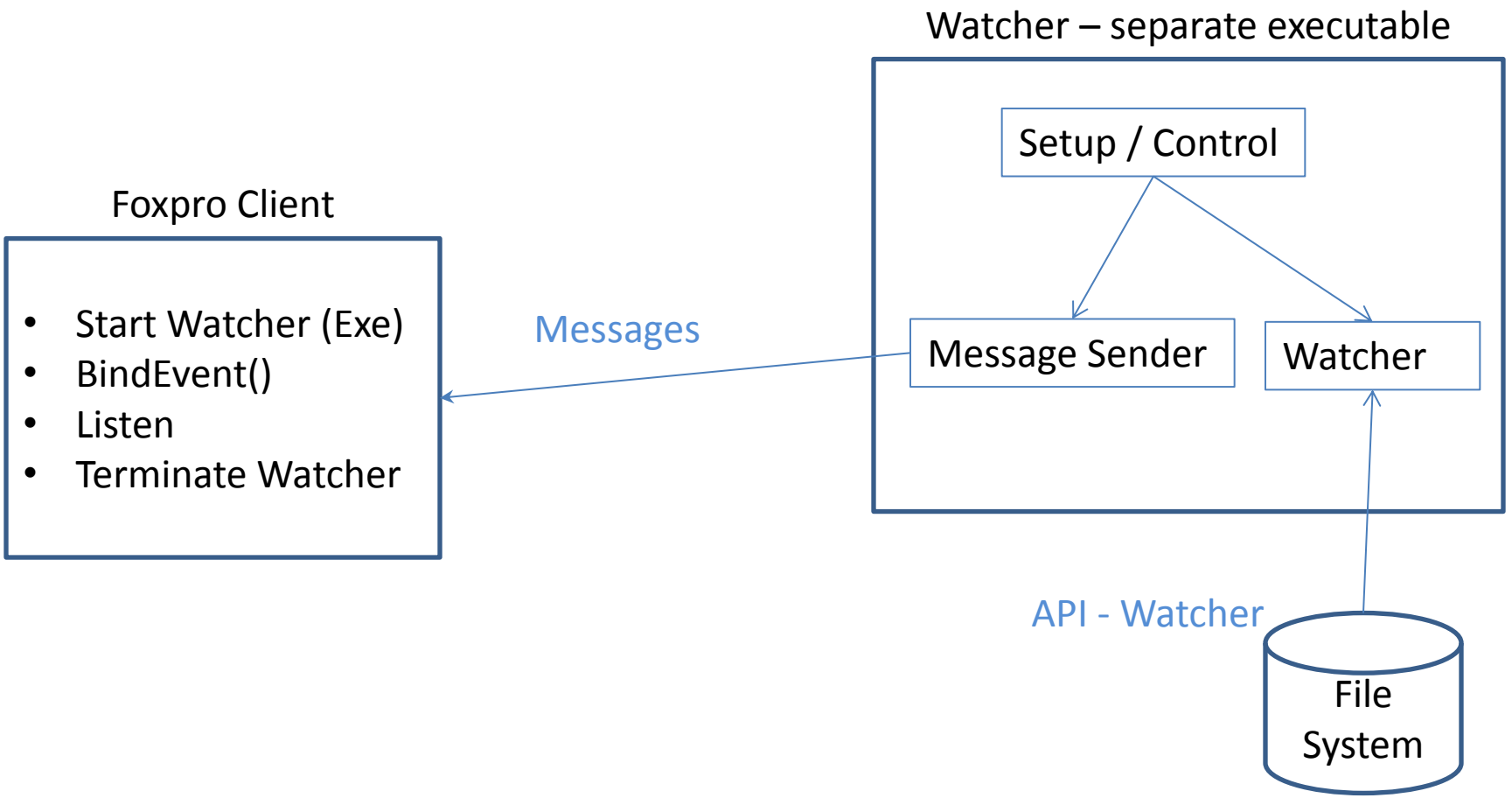
Choices (next)

1. Messages
2. API - Watcher

Where are we

- Contents
  - Purpose
  - Possibilities
  - Choices
    - Implementation
    - Messaging
      - Blocking vs Non-blocking
      - Parameters
    - Detect changes API
    - Program flow

Implementation



Where are we

- Contents
  - Purpose
  - Possibilities
  - Choices
    - Implementation
    - Messaging
      - Blocking vs Non-blocking
      - Parameters
    - Detect changes API
    - Program flow

Choice – Messages
Blocking / Non-blocking

Foxpro can bind to windows messages

- PostMessage() – Non-blocking
- SendMessage() – Blocking

```
BOOL WINAPI PostMessage(
  _In_opt_  HWND hWnd,
  _In_      UINT Msg,
  _In_      WPARAM wParam,
  _In_      LPARAM lParam
);
```

Choice – Messages

Parameters – PostMessage/SendMessage()

```
BOOL WINAPI PostMessage(
  _In_opt_  HWND hWnd,
  _In_      UINT Msg,
  _In_      WPARAM wParam,
  _In_      LPARAM lParam
);
```

- Int hnwd (specific or broadcast)
- Int messageId
  - 0x0000 – 0x03ff : System reserved
  - 0x0400 – 0x7fff : Private window classes
  - 0x8000 – 0xbfff : Private messages
  - 0xc000 – 0xffff : via RegisterWindowMessage()
- Int wParam
- Int lParam

There's a problem with sending strings -filename

Choice – Messages Parameters needed

- Number of parameters
  1. What happened (new/delete/change/rename)
  2. File name
  3. New filename (rename)
- Type
  - What happened (integer)
  - File names (string)

```
BOOL WINAPI PostMessage(
  _In_opt_  HWND hWnd,
  _In_      UINT Msg,
  _In_      WPARAM wParam,
  _In_      LPARAM lParam
);
```

Choice – Messages

Parameters – PostMessage/SendMessage

```
BOOL WINAPI PostMessage(
  _In_opt_  HWND hWnd,
  _In_      UINT Msg,
  _In_      WPARAM wParam,
  _In_      LPARAM lParam
);
```

- Sending Strings – Blocking
- Second parameter – WM_COPYDATA
- Third parameter – hwnd of the sender
- Fourth Parameter – pointer to a struct that contains the data
- There's a standard timeout of 5 seconds
  - Can be changed with **SendMessageTimeout()**

Choice – Messages

Parameters – PostMessage/SendMessage()

- Sending strings – Non-blocking
- Strings have to be allocated and Released
  - SHParseDisplayName() / SHILCreateFromPath()
    - Convert a filename to an integer
    - Is said to be slow (filesystem lookup)
    - Space must be released (CoTaskMemFree())
    - Hard to use if a filename does not exist (delete/rename)
  - Allocate a global buffer – divide in slots
  - GlobalAddAtom(string) – **GlobalDeleteAtom**
    - 16 bit Id
    - Table of 0x4000 (16384) items

Choice – Messages - Strings

- Blocking
  - easy with WM_COPYDATA
- Non-Blocking
  - Overhead of allocating space
  - Overhead of releasing space
  - When to release space ?
- Choice:Blocking (SendMessage with WM_COPYDATA)
  - Simple
  - No overhead of allocating/releasing

Choice – Messages - WM_COPYDATA

- #define WM_COPYDATA 0x004A
- COPYDATASTRUCT // lParam
  - ULONG_PTR dwData; // integer
  - DWORD cbData; // length of data in lpData
  - PVOID lpData; // pointer to the data

```
BOOL WINAPI PostMessage(
  _In_opt_  HWND hWnd,
  _In_      UINT Msg,
  _In_      WPARAM wParam,
  _In_      LPARAM lParam
);
```

Choice – WM_COPYDATA - Implementation

- COPYDATASTRUCT // lParam
  - ULONG_PTR dwData; // integer
  - DWORD cbData; // length of data in lpData
  - PVOID lpData; // pointer to the data
- dwData
  - 1-2 : length of filename1
  - 3-4 : length of filename2
- lpData
  - 1-1 : new/delete/modify/rename
  - 2- : filename1 + filename2

```
BOOL WINAPI PostMessage(
  _In_opt_  HWND hWnd,
  _In_      UINT Msg,
  _In_      WPARAM wParam,
  _In_      LPARAM lParam
);
```

Choice – WM_COPYDATA - Implementation

```
local dwdata, cbdata , lpData, s, action
local len1, len2, filename1

dwdata = ctobin(Sys(2600, lparam+0, 4), '4rs')
cbdata = ctobin(Sys(2600, lparam+4, 4), '4rs')
lpData = ctobin(Sys(2600, lparam+8, 4), '4rs')
s= sys(2600, lpData, cbdata )

action = asc(m.s)
len1 = bitrshift(m.dwdata, 16)
len2 = bitand(m.dwdata, 0x0000ffff)
filename1 = substr(m.s, 2, m.len1)

do case
case m.action == FSW_FILE_ACTION_ADDED
    =m.this.File_Added(m.filename1)

case m.action == FSW_FILE_ACTION_REMOVED
    =m.this.File_Deleted(m.filename1)

case m.action == FSW_FILE_ACTION_MODIFIED
    =m.this.File_Modified(m.filename1)

case m.action == FSW_FILE_ACTION_RENAMED
    =m.this.File_Renamed( ;
                m.filename1, ;
                substr(m.s, 2+m.len1, m.len2) ;
                )

otherwise
    assert false
endcase
```

- dwData
  - 1-2 : length of filename1
  - 3-4 : length of filename2
- lpData
  - 1-1 : new/delete/modify/rename
  - 2- : filename1 + filename2

Where are we

- Contents
  - Purpose
  - Possibilities
  - Choices
    - Implementation
    - Messaging
    - Detect changes API
    - Program flow

API

1. SHChangeNotifyRegister() API (in Solutions/Europa/WindowsEvents)
2. FindFirstChangeNotification() and FindNextChangeNotification() API
3. ReadDirectoryChangesW() API

API – SHChangeNotifyRegister()

- In Solutions/Europa/WindowsEvents
- Detects changes
- More or less reliable for local drives
- Could not get it to work properly on a mapped drive (only Updated file/directory – Server 2012)

API
FindFirstChangeNotification()
FindNextChangeNotification()

- Tells that something in a directory has changed
- But does not tell **what** has changed

API – ReadDirectoryChangesW()

- Most reliable
- Unicode – No Ansi version
- Construct possible to watch multiple directories simultaneously

API – ReadDirectoryChangesW()

```
BOOL WINAPI ReadDirectoryChangesW(
    _In_       HANDLE hDirectory,                - CreateFile() with async IO
                                                       FILE_FLAG_OVERLAPPED
    _Out_      LPVOID lpBuffer,                  - buffer receiving the data (*)
    _In_       DWORD nBufferLength,              - length of buffer
                                                  Max 64k for network drives
    _In_       BOOL bWatchSubtree,               - watch subdirectories ?
    _In_       DWORD dwNotifyFilter,             - changes to watch (*)
    _Out_opt_ LPDWORD lpBytesReturned,           - bytes returned
    _Inout_opt_  LPOVERLAPPED lpOverlapped,            - for async IO
    _In_opt_   LPOVERLAPPED_COMPLETION_ROUTINE lpCompletionRoutine
                                                       - not used - callback

    );
    (*) next page(s)
```

API – ReadDirectoryChangesW()

DWORD dwNotifyFilter

FILE_NOTIFY_CHANGE_FILE_NAME	Rename, create, or delete a file
FILE_NOTIFY_CHANGE_DIR_NAME	Rename, create, or delete a directory
FILE_NOTIFY_CHANGE_ATTRIBUTES	Any attribute change in the watched directory or subtree
FILE_NOTIFY_CHANGE_SIZE	Change size (1)
FILE_NOTIFY_CHANGE_LAST_WRITE	Change to the last write-time (1)
FILE_NOTIFY_CHANGE_LAST_ACCESS	Any change to the last access time (2)
FILE_NOTIFY_CHANGE_CREATION	Any change to the creation time
FILE_NOTIFY_CHANGE_SECURITY	Any security-descriptor change

(1) Only when disk cache is sufficiently flushed

(2) Don’t monitor a whole drive with **FILE_NOTIFY_CHANGE_LAST_ACCESS**

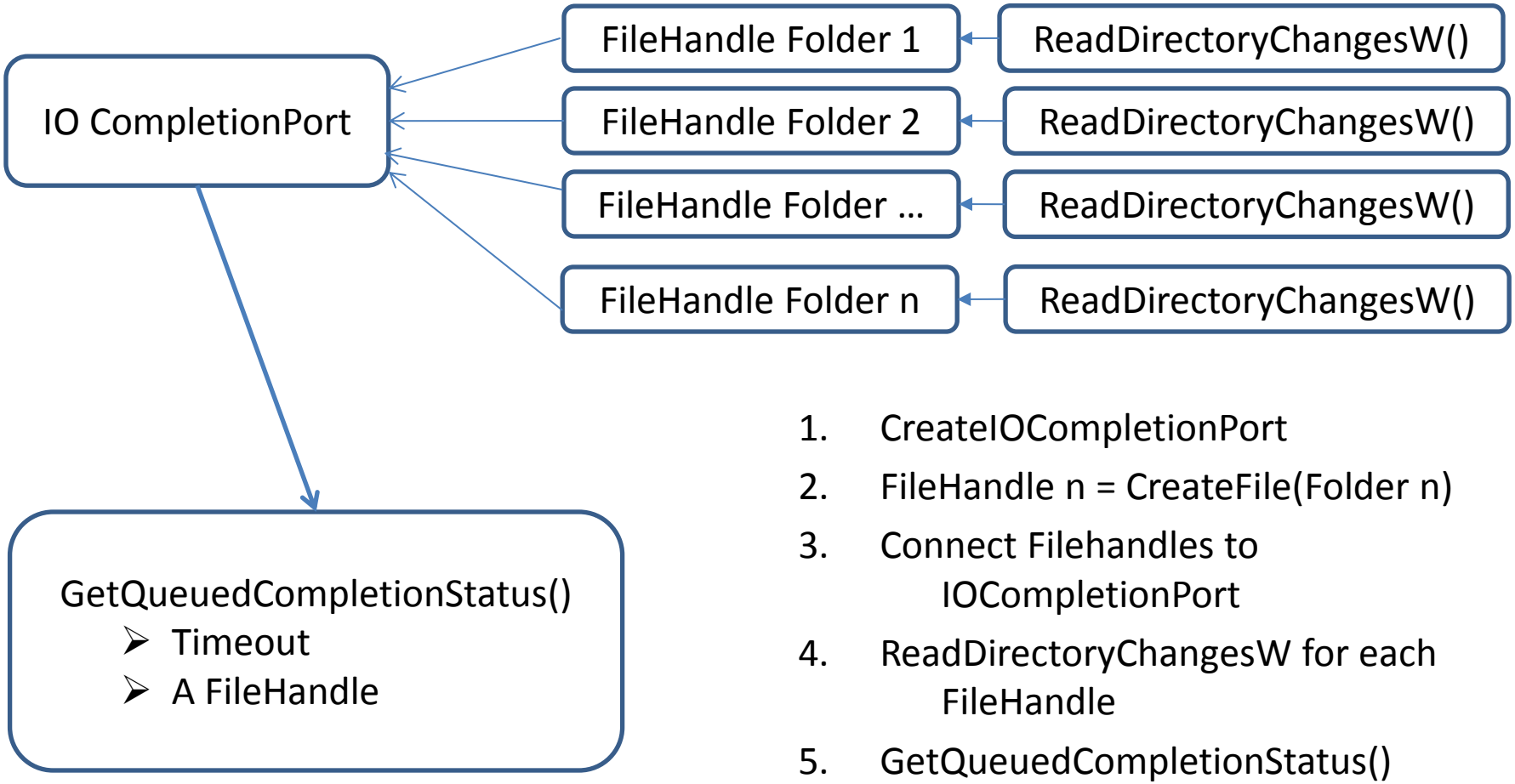
API – ReadDirectoryChangesW()

Data returned in lpBuffer

```
typedef struct _FILE_NOTIFY_INFORMATION
{       DWORD NextEntryOffset;
            The number of bytes that must be skipped to get to the
            next record. A value of zero indicates that this is the last record
        DWORD Action;
            Added
            Removed
            Modified
            Rename_old_file
            Rename_new_file
        DWORD FileNameLength;
        WCHAR FileName[1];
            Variable-length field that contains the file name
            Unicode – not null terminated
}
```

API – ReadDirectoryChangesW()

Async IO



1. CreateIOCompletionPort
2. FileHandle n = CreateFile(Folder n)
3. Connect Filehandles to IOCompletionPort
4. ReadDirectoryChangesW for each FileHandle
5. GetQueuedCompletionStatus()
   - Timeout
   - FileHandle

API – IOCompletionPort()

```
HANDLE WINAPI CreateIoCompletionPort(
        _In_     HANDLE FileHandle,
                  An open file handle or INVALID_HANDLE_VALUE.
        _In_opt_  HANDLE ExistingCompletionPort,
                  A handle to an existing I/O completion port or NULL
        _In_     ULONG_PTR CompletionKey,
                  Any Id
        _In_     DWORD NumberOfConcurrentThreads
);
```

API – GetQueuedCompletionStatus()

```
BOOL WINAPI GetQueuedCompletionStatus(
            _In_   HANDLE CompletionPort,
            _Out_  LPDWORD lpNumberOfBytes,
             _Out_  PULONG_PTR lpCompletionKey,
            _Out_  LPOVERLAPPED *lpOverlapped,
             _In_   DWORD dwMilliseconds
);
```

Where are we

- Contents
  - Purpose
  - Possibilities
  - Choices
    - Implementation
    - Messaging
    - Detect changes API
    - Program flow

Program flow

- Setup
- Loop

Program flow - Setup

- For each folder to watch
  - FileHandle = CreateFile(…, OPEN_EXISTING, FILE_OVERLAPPED)
- Create IOCompletionPort
- For each FileHandle
  - Attach FileHandle to IOCompletionPort
- For each FileHandle
  - Start ReadDirectoryChanges()
- State = STATE_LISTEN

Program flow - Loop

Do while not done

case State = STATE_LISTEN

GetQueuedCompletionStatus() with timeout
if(there is data) State = STATE_PROCESS_DATABUFFER

case State = STATE_PROCESS_DATABUFFER

Attach other buffer (using two buffers)

Start ReadDirectoryChangesW()

Process buffer (Entries + SendMessage)

State = STATE_LISTEN

References

- CreateFile
- ReadDirectoryChangesW https://msdn.microsoft.com/en-us/library/windows/desktop/aa365465(v=vs.85).aspx
- CreateIoCompletionPort https://msdn.microsoft.com/en-us/library/windows/desktop/aa363862(v=vs.85).aspx
- GetQueuedCompletionStatus https://msdn.microsoft.com/en-us/library/windows/desktop/aa364986(v=vs.85).aspx
- Understanding ReadDirectoryChangesW() Part 1 and 2
  - http://qualapps.blogspot.be/2010/05/understanding-readdirectorychangesw.html

On AtoutFox soon