Win32 Threads

Edwin Keville 13718661

Win32 API

History and evolution

The Win32 API was first introduced alongside Windows NT 3.1. It succeeded Win16 which accompanied windows 3.1 and older. While it was available earlier, it was mainly the extremely popular Windows 95 operating system that resulted in its widespread use and a great many applications being ported over to it.

Windows, as the name would suggest, introduced a window based graphical environment with various applications running concurrently. So while initially many thought there would not be much need for multiprocessing on personal computers this new way of working showed that it was in fact essential.

Win16 did offer multiprocessing but it was extremely limited. This limited environment however provided a lot of the logic and methods that would later be implemented on the more advanced API’s which followed.

The first attempts from windows at multiprocessing were OS/2 with Presentation Manager. But the serialized inputs from elements such as the keyboard and mouse meant that one application could possibly tie up the entire system. Deserializing the messaging queue required a new fix, and this fix was multithreading.

While every new version of windows makes additions to the API, the naming remains mostly the same. Win16 was followed by Win32, with minor variations(Win32s and Win32c). and these were followed by Win64

For the purpose of our case study we will be looking mainly at Win32.

Threads

Threads themselves are a basic unit of CPU utilization and comprise of a thread ID, a program counter. A register set and a stack. They are a path of execution and every process requires at least one thread. They exist mainly on two levels, the user level and the Kernel level. The win32 API operates on a kernel level, meaning it is handled directly by the operating system.

There is a close link between user and kernel level threads and Win32 uses a one-to-one model whereby each user thread is linked to a kernel thread. This allows for concurrency as if one thread makes a blocking system call, the other threads may still run. Although this method does mean that for each thread there must be a corresponding kernel thread. This overhead means that Win32 restricts the amount of threads supported.

The Win32 library also exists within the Kernel and contains all the information such as code and data structures that the programmer must use.

Programs are mostly written in the C/C++ language and the threads themselves are representations of functions. A program begins execution with its main thread and new threads can be created from within.

The API function which creates a new thread is called CreateThread.

For example

hThread = CreateThread(&security\_attributes, dwStackSize, ThreadProc, pParam, dwFlags &idThread);)

security\_attributes: these are just that. The security attributes for the thread. This can be set to null in windows NT and is ignored altogether in windows 98. It is a pointer to a structure and setting to null will give it the default security values.

dwStacksize: will declare the initial size of the stack size in bytes although windows can dynamically expand this when it is required.

threadProc: is a pointer to the thread function

pParam: is the main argument for threadProc and is how the threads can share data between main and secondary threads.

dwFlags: Here a flag can be used such as CREATE\_SUSPENDED. If this happens then the thread is created in a suspended state and will remain so until ResumeThread is called.

&idThread: this is simply a pointer to a variable that will hold the ID for the thread.

in the process.H header file there is also \_beginthread. This is much simpler and works perfectly fine for most applications

hThread = \_beginthread (ThreadProc, uiStackSize, pParam)

Threads such as this are then ended using the \_endthread function. When CreateThread is used, it is recommended to close with ExitThread otherwise deadlocks may occur.

Thread priorities are also very important when it comes to handling and they come in 5 variations: Highest, Above Normal, Normal, Below Normal and Idle

(eg: THREAD\_PRIORITY\_HIGHEST, THREAD\_PRIORITY\_ABOVE\_NORMAL…)

HANDLE is specific to win32 and cannot be found on Linux. It differs from the thread ID in that it is a token which allows you to perform tasks such as waiting for the thread or to kill it.

Looking at a code sample can give us a better idea.

Void ThreadProc(void \*param);

Int main()

{

int n;

int i;

int val = 0;

HANDLE handle;

Printf(“Please enter the number of threads required: ”)

Scanf(“%d”, &n);

For(i=0;i<=n;i++){

val = I;

//here the thread is created

handle = (HANDLE) \_beginthread( ThreadProc, 0, &val);

WaitForSingleObject(handle, INFINITE);

}

return 0;

}

void ThreadProc(void \*param){

int h = \*((int\*)param)

printf(“%d thread is Running”, h);

\_endthread();

}

this simple program serves to create more threads within the main thread. The WaitForSingleObject function will wait for the completion of another thread. As shown here the parameter passed to it is the handle.

//scratch pad

Benefit of threads responsiveness/resource sharing/economy/utilization of multiprocessor architectures.

Effective use of threading? Not just using threads all the time.. but knowing when to use them. The 1/10th rule. Memory takes longer to read than 1/10th of a second so use a thread?? NEIN!!

MFC is a wrapper for threads in win32

The WINE project is a wrapper which allows unix based systems (such as mac) to run windows based applications.

Posix-32 is is like a windows version of wine

Posix threads

Mutex

POSIX THREADS

Initially hardware vendors would create their own versions of threads depending on their needs for a specific project. Of course this caused problems for developers as it greatly reduced the portability of applications. The solution to this as to create a standard. This was specified by the IEEE POSIX 1003.1c standard(1995) for UNIX systems and implementations which adhered to the standard were referred to as POSIX thread or Pthreads.

This standard has continued to grow and evolve with todays programming. The API itself is implemented using the pthreads.h header and is a set of C language programming types and procedure calls.