Systems Programming under Windows

Dmitri Loguinov (ver 1.11, Spring 2022)

The purpose of this document is to refresh common C/C++ programming techniques, introduce MSDN notation, and set forth basic ideas for effective debugging in Windows.

1. MSDN and Visual Studio

MSDN notation may first appear unusual for those mostly familiar with Unix; however, it is quite simple to learn. This section outlines the basic steps in tackling MSDN and overviews the main elements you should know.

1.1 General Ideas

The first step is to understand how to search MSDN (http://msdn.microsoft.com), which is also known as Microsoft Docs. To avoid frustration, you should be prepared that certain functions (e.g., `select`) produce many irrelevant results from .NET, C#, SQL, and other areas. One approach is to allow Google to pick the specific page based on the frequency of request from other users and its internal ranking algorithm (e.g., by searching for "MSDN select c++"). Another approach is to type the desired function in Visual Studio (any .cpp or .h file), place the cursor within its name, and press F1 to bring up the relevant documentation. Finally, you can manually scroll though MSDN search results until you see the version with "(Windows)" in the title (e.g., "select function (Windows)").

Reading MSDN and understanding the various caveats will save you plenty of time when an APIs crashes because you are passing NULL pointers to it or incorrectly handling its response. Part of this understanding involves knowing the various datatypes that Microsoft uses. It is recommended that you perform a lookup on every unfamiliar type you encounter. Below are some of the most common datatypes:

```
CHAR = char
DWORD = unsigned int          // "double word"
BYTE = unsigned char
BOOL = int                    // boolean
HANDLE = void*
LONG = int
UINT = unsigned int
UINT64 = unsigned __int64
```

Many pointer types are created from existing scalar types by prepending them with `LP` (long pointer):

```
LPDWORD = DWORD*
LPVOID = void*
LPCSTR = char* (expects NULL termination, C-style string)
```