System Programming Homework #2

2018440059 박재선

# 사용된 자료형, 함수 조사

1. 오류 해결
   1. char, wchar\_t,,

char 형식은 ASCII 문자 혹은 ISO-8859의 문자를 입력하는데 사용한다. 일반적으로 Microsoft 에서는 8비트 형식이다. wchar\_t는 char의 확장판으로, 16비트 문자인 유니코드를 나타내기 위해 만들어졌다. winnt.h 에서 각각 CHAR와 WCHAR로 typedef되어 있다.

|  |
| --- |
| winnt.h |
|  |

* 1. LPSTR, LPWSTR(winnt.h)

LPSTR은 char로 이루어진 문자열, LPWSTR은 wchar\_t로 이루어진 문자열이다.

|  |
| --- |
| winnt.h |
|  |

* 1. LPCSTR, LPCWSTR,

LPCSTR은 const char \* 상수형 문자에 대한 포인터이다. LPCWSTR은 const wchar\_t \* 상수형 와이드문자에 대한 포인터이다.

|  |
| --- |
| winnt.h |
|  |

* 1. TCHAR, LPTSTR, LPCTSTR

전처리기로 시스템의 지정된 상태에 따라 TCHAR, LPTSTR, LPCTSTR를 지정하도록 되어있다.

현재는 WCHAR를 기준으로 TCHAR, LPTSTR, LPCTSTR가 지정되어 있다.

|  |
| --- |
| winnt.h |
| wchar\_t →TCHAR  wchar\_t \*→ LPTSTR  const wchar\_t \* →LPCTSTR    char →TCHAR  char \* →LPTSTR  const char \* →LPCTSTR |

* 1. 오류

텍스트이(가) 표시된 사진

자동 생성된 설명

매크로 함수 TEXT(...)의 반환형은 const wchar\_t\*인데 ErrorHandler의 매개변수는 LPTSTR형식이다. LPTSTR는 const char\* 이므로 호환되지 않는다. ErrorHandler 함수 내에서 어떻게 매개변수가 사용되는지 살펴보자.

텍스트이(가) 표시된 사진

자동 생성된 설명

문자열로써 사용되고 있다. 문자열에 접근하여 조작하는 연산이 없어서, 매개변수의 타입을 LPCTSTR로 지정해도 문제없을 것 같다. 실제로 중간에 LPTSTR형식의 lpszFunction을 LPCTSTR로 강제 형변환해서 사용하는 모습이 포착되었다. 매개변수 타입을 LPCTSTR로 바꿔주고 강제 형변환을 지워주자.

텍스트이(가) 표시된 사진

자동 생성된 설명

1. HeapAlloc (heapapi.h)



|  |
| --- |
| DECLSPEC\_ALLOCATOR LPVOID HeapAlloc(  [in] HANDLE hHeap,  [in] DWORD dwFlags,  [in] SIZE\_T dwBytes ); |

* 1. HANDLE GetProcessHeap()

메모리가 할당될 힙의 핸들을 얻어온다. 현재 포르세스의 힙의 핸들을 반환한다. HeapCreate()를 이용할 수도 있다. HeapCreate는 현재 프로세스 소유의 힙의 핸들을 생성한다.

* 1. HEAP\_ZERO\_MEMORY (winnt.h)

할당한 메모리 영역을 0으로 초기화 하라는 옵션이다. 이외에도 다른 옵션들을 지정할 수 있다. 이는 힙의 생성 시에 지정된 옵션과 다른 옵션으로 동적할당 받고 싶을 때 사용한다.

텍스트이(가) 표시된 사진

자동 생성된 설명

1. LocalAlloc (WinBase.h)

힙에서 시정된 바이트 수만큼의 메모리를 할당한다. 로컬 함수(LocalAlloc 등)은 다른 오버헤드가 크고 기능이 적어서 되도록 힙 함수(HeapAlloc 등)을 사용해야 한다.



* 1. Local Memory Flags

코드에서 사용한 LMEM\_ZEROINIT은 할당한 메모리를 0으로 초기화 하는 옵션이다. 그밖에도 다양한 옵션들이 존재한다. 더이상 사용되지 않아 입력해도 무시되는 플래그도 존재한다.

텍스트이(가) 표시된 사진

자동 생성된 설명

1. strsafe.h

버퍼와 관련된 연산을 할 때 보안 문제가 발생할 수 있다. 런타임 중에 안전하고 적절한 버퍼 처리를 위한 함수들을 제공한다. 문자열 만들기, 문자열 연결하기, 문자열 길이 세기 등의 기능을 제공한다. char단위, byte 단위 두 가지 종류의 연산을 제공한다. 아래는 코드에서 사용한 함수들에 대한 설명이다.

* 1. StringCchPrintf



|  |
| --- |
| STRSAFEAPI StringCchPrintfA(  [out] STRSAFE\_LPSTR pszDest,  [in] size\_t cchDest,  [in] STRSAFE\_LPCSTR pszFormat, ...); |

destination 문자열의 형식은 LPTSTR이어야 한다. 현재 LPTSTR은 wchar\_t\* 로 정의되어 있다. 포맷을 지정한다. 포맷 문자열의 형식은 LPCTSTR이어야 한다.

* 1. StringCchLength



|  |
| --- |
| STRSAFEAPI StringCchLengthA(  [in] STRSAFE\_PCNZCH psz,  [in] size\_t cchMax,  [out] size\_t \*pcchLength ); |

인자에 LPCTSTR (const wchar\_t \*)형식의 상수문자열을 받아서 문자열의 길이를 잰다. cchMax에서 문자열의 최대길이를 입력받는다. 버퍼의 크기를 입력하면 된다. 정상적인 문자열이라면 문자열의 길이는 버퍼의 크기보다 작거나 같다.

1. Writeconsole (consoleapi.h)



|  |
| --- |
| consoleapi.h |
|  |

출력한 콘솔의 핸들과, 출력할 문자열, 문자열의 길이를 입력으로 받고, 출력된 문자열의 길이와 NULL값을 입력받는다.

첫 인자인 콘솔의 핸들은 코드에서는 표준출력의 핸들을 가져왔다. 이 핸들은 GetStdHandle (STD\_OUTPUT\_HANDLE)을 이용해 가져온다.

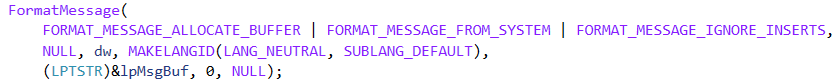
출력인 lpNumberOfCharsWritten (출력된 문자열의 길이)는 정상적인 경우 입력한 문자열의 길이와 같지만, 출력 중 에러가 발생한 경우 문자열을 모두 출력하지 못하면 입력 문자열의 길이보다 작을 수 있다.

1. ErrorHandler



예제 코드에서는 매개변수의 타입이 LPTSTR이었지만 위에서 언급한 문제로 인해 LPCTSTR로 바꾸었다.

* 1. FormatMessage (WinBase.h)



|  |
| --- |
| winbase.h |
| DWORD FormatMessage(  [in] DWORD dwFlags,  [in, optional] LPCVOID lpSource,  [in] DWORD dwMessageId,  [in] DWORD dwLanguageId,  [out] LPTSTR lpBuffer,  [in] DWORD nSize,  [in, optional] va\_list \*Arguments ); |

* + 1. Flags

FORMAT\_MESSAGE\_ALLOCATE\_BUFFER: 포맷된 메시지를 모두 담을 수 있는 버퍼를 할당하라는 플래그이다.

FORMAT\_MESSAGE\_FROM\_SYSTEM: 시스템 메시지 테이블 리소스에서 요청된 메시지를 찾아야 한다는 플래그이다.

FORMAT\_MESSAGE\_IGNORE\_INSERTS: 메세지 내 %1, %2과 같은 삽입 시퀀스를 무시한다.

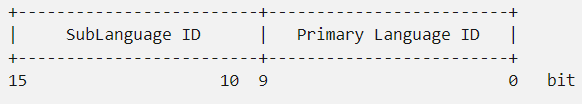
Flag는 lpSource를 해성하는 방법과 포맷 옵션을 지정한다.

예제코드에서는 lpSource가 NULL로 입력된다.

* + 1. dw = GetLastError();

마지막 에러의 ID를 가져와서 FormatMessage에 전달한다.

* + 1. MAKELANGID(LANG\_NEUTRAL, SUBLANG\_DEFAULT) (winnt.h)



위와 같은 포맷으로 언어 id를 구성해준다. FormatMessage에 이 언어 id를 입력해 주면, 메세지를 해당 언어에 맞게 번역한다.

에러코드를 분석하여 출력 메세지를 만들어서 lpMsgBuf에 넣어준다.

FORMAT\_MESSAGE\_ALLOCATE\_BUFFER flag가 지정되었기 때문에 버퍼의 크기를 알아서 할당한다. 따라서 nSize는 0으로 입력한다.

* 1. MessageBox MessageBoxW (WinUser.h)



위의 FormatMessage로 에러를 나타내는 문자열을 만들었다. 그 내용을 시스템 다이얼로그 창으로 띄워 보여준다.

|  |
| --- |
| int MessageBox(  [in, optional] HWND hWnd,  [in, optional] LPCTSTR lpText,  [in, optional] LPCTSTR lpCaption,  [in] UINT uType ); |

hWnd가 NULL이면 다이얼로그 창의 소유창은 없다. 종속되지 않는다.

lpText는 다이얼로그의 내용이다.

lpCaption은 다이얼로그의 제목이다.

uType은 다이얼로그 박스의 내용과 동작이다. MB\_HELP, MB\_YESNO등의 옵션이 있다. MB\_OK면 다이얼로그에 ok버튼이 하나 있고, MB\_YESNO면 yes, no 버튼이 존재한다.

# 코드 분석 (코드 전문)

|  |
| --- |
| MyData구조체를 커스텀할 수 있다.  main함수      스레드 함수    the content of thread 아래의 내용을 커스텀할 수 있다.  에러 핸들러 함수  매개변수를 LPTSTR 에서 LPCTSTR로 바꿔주었다. 그리고 bufLen이라는 변수에 생성할 문자열(lpDisplayBuf)의 길이를 저장했다. |

MyData라는 구조체가 있다. struct MyData는 MYDATA, strunc MyData\* 는 PMYDATA로 typedef되어 있다. 이 구조체가 스레드의 인자로 들어가게 된다. 스레드는 메인함수에서 MAX\_THREADS개 만큼 생성된다. 모든 스레드가 종료할 때까지 대기하고, 모든 스레드가 종료하면 스레드의 핸들과 스레드의 인자로 사용된 MYDATA객체를 반환한다.

스레드의 동작 함수는 매개변수의 인자로 전달받은 MYDATA객체의 정보를 프린트하는 일을 한다. 스레드의 동작은 필요에 따라 변경할 수 있다.

CreateThread가 실패하면 ErrorHandler라는 함수를 호출한다. 이 함수에서는 GetLastError()함수를 이용하여 최근에 발생한 오류를 탐색하여 오류 내용을 다이얼로그로 표시한다.

|  |  |  |
| --- | --- | --- |
| 출력1 | 출력2 | 출력3 |
|  |  |  |

각 스레드는 전달받은 인자의 정보를 출력한다. 이 인자는 main함수 내에서 지정되며 스레드의 생성 순서대로 {0,100}, {1,101}, {2,102}가 입력된다.

스레드의 동작 순서는 정해지지 않았기 때문에 실행할 때마다 출력이 다르다.