WinDBG 실무

주식회사 하제소프트
(www.hajesoft.co.kr)

강사 이 봉 석

과정소개

- 윈도우 응용프로그램, 윈도우 서비스프로그램, 윈도우 디바이스 드라이버를 개발하는 개발자들로 하여금 고급디버깅기술을 제공하는 '윈도우 디버거(WinDBG)' 사용방법을 익히게 하여, 고급 시스템 프로그래머를 양성하는데 있습니다
- 윈도우 디버거(WinDBG)를 사용하는 개발자는 실무에서 고급 시스템 프로그래머가 갖추어야 할 중요한 디버깅지식을 습득함과 동시에 시간과 비용을 최대한 아끼는 프로그래밍 습관과 우수한 결과물을 만들어 낼 수 있습니다

강사약력

- 본 강사(이봉석)는 기업교육(삼성첨단기술센터, LG러닝센터, MDS테크놀러지)과 사설교육등에서 13년 강사 이력을 가지고 있으며, 현재 주식회사 하제소프트 대표이사를 역임하고 있습니다.
- 저서로는, “윈도우 디바이스 드라이버”, “실전 윈도우 디바이스 드라이버”, “고급개발자들만이 알고 있던 디바이스 드라이버 구조와 원리, 그리고 제작 노하우”, “실전 윈도우 CE 가이드”, “USB 너뭐니”가 있습니다.
- 주식회사 하제소프트는 17년간 업체들이 필요로 하는 윈도우 디바이스 드라이버, 리눅스 디바이스 드라이버 그리고 훰웨어를 개발, 제공하는 업체입니다.

과정 진행 목차

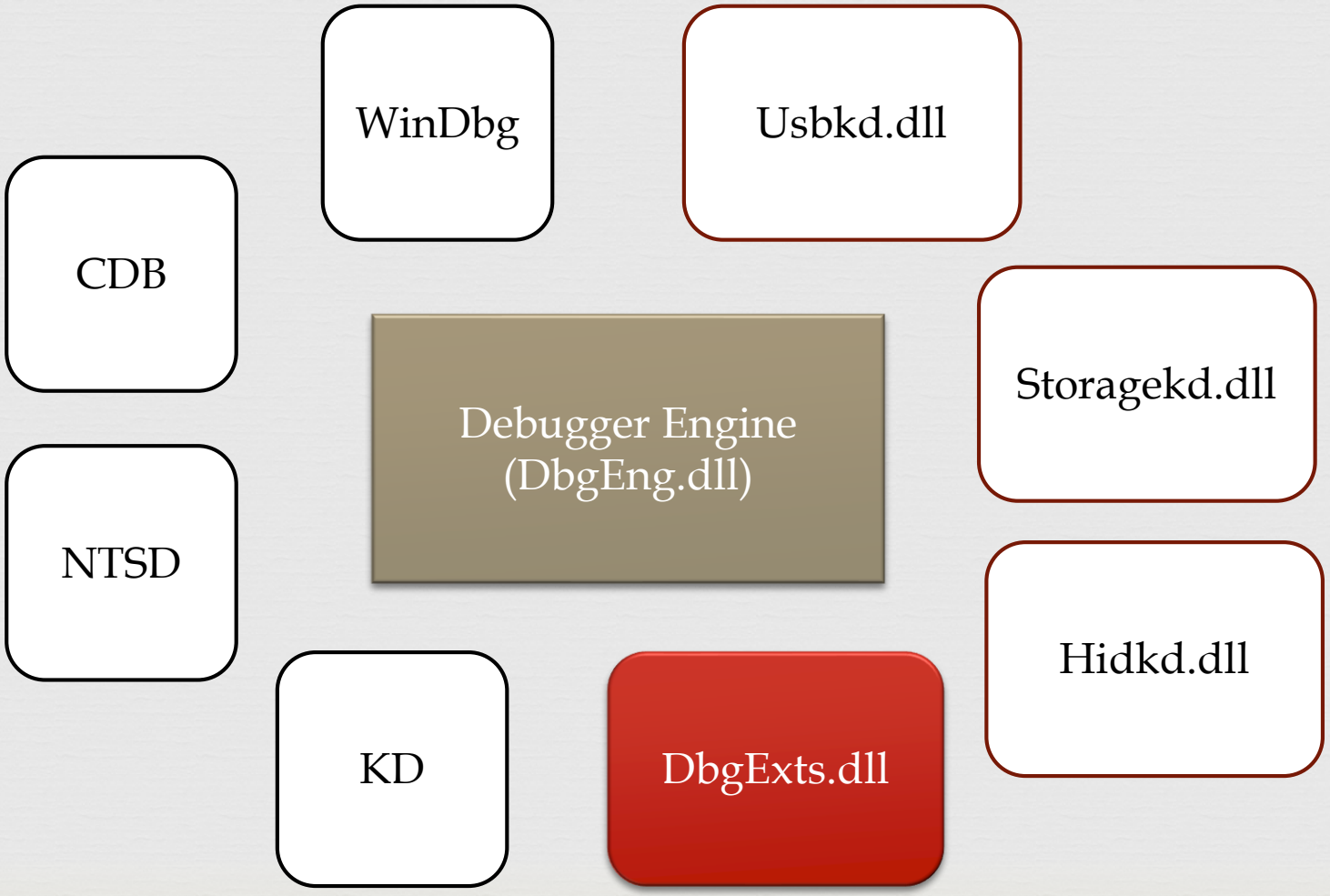
- **WinDBG 환경 준비**
  - Visual Studio 2015, WDK(Windows Driver Kit), WinDBG
  - WinDBG 심볼설정
- **마이크로 프로세서 Calling 규약**
  - x86 CPU에서 C 함수 호출과 어셈블리어 연관성 분석
  - x64 CPU 에서 C 함수 호출과 어셈블리어 연관성 분석
- **WinDBG 로칼 디버깅**
  - 응용프로그램, 서비스프로그램 디버깅
- **BSOD(BlueScreen Of Death) 커널 덤프파일**
  - 커널 덤프파일 생성 설정및 덤프분석
- **WinDBG 원격 디버깅**
  - 서비스프로그램, 커널 디바이스 드라이버 디버깅
- **하드웨어와 관련된 WinDBG 주요 명령어**
  - PCI, USB 버스를 사용하는 하드웨어를 다룰때, 유용한 WinDBG 확장명령어를 소개합니다
- **WinDBG 확장 모듈**
  - 확장기능을 제공하는 모듈 구현

WinDBG 확장모듈

- WinDBG 확장모듈을 개발하여 자신만의 명령어를 지원하는 방법을 배워봅니다.
- COM(Component Object Model) 을 이해해야 합니다

1. Debugger Engine

- Debugger Extension 라이브러리와 Debugger 응용프로그램을 개발할 수 있도록 지원합니다



2. Debugger Engine API

- Debugger Engine 이 제공하는 API를 살펴봅니다

유형	의미
Client Functions	COM Interface 를 얻어옵니다
Debug Engine Interfaces	일반적인 Interface를 포함합니다
Callback Debug Engine Interfaces	Callback 관련 Interface를 포함합니다
Other Debug Engine Interfaces	BreakPoint, Symbol Group관련 Interface를 포함합니다

2.1 Client Functios

- 새로운 Client 객체를 생성하고 COM Interface 를 얻어옵니다

함수	의미
HRESULT DebugConnect(_In_ PCSTR RemoteOptions, _In_ REFIID InterfaceId, _Out_ PVOID *Interface);	원격 호스트에 연결하고, 새로운 Client 객체를 생성하고 COM Interface를 얻어옵니다
HRESULT DebugCreate(_In_ REFIID InterfaceId, _Out_ PVOID *Interface);	새로운 Client 객체를 생성하고 COM Interface를 얻어옵니다

```
IDebugClient *          DebugClient;
PDEBUG_CONTROL    DebugControl;
HRESULT    Hr;

if ((Hr = DebugCreate(__uuidof(IDebugClient), (void **)&DebugClient)) != S_OK)
{
            return Hr;
}
Hr = DebugClient->QueryInterface(__uuidof(IDebugControl), (void **)&DebugControl);
```

2.2 Debug Engine Interfaces

- 일반적인 Interface 들을 포함합니다

기본 Interface	관련된 Interfaces
IDebugAdvanced	IDebugAdvanced2, IDebugAdvanced3
IDebugClient ✓	IDebugClient2, IDebugClient3, IDebugClient4, IDebugClient5
IDebugControl ✓	IDebugControl2, IDebugControl3, IDebugControl4, IDebugControl5, IDebugControl6, IDebugControl7
IDebugDataSpaces	IDebugDataSpaces2, IDebugDataSpaces3, IDebugDataSpaces4,
IDebugRegisters	IDebugRegisters2
IDebugSymbols ✓	IDebugSymbols2, IDebugSymbols3
IDebugSymbolObjects	IDebugSymbolObjects2, IDebugSymbolObjects3, IDebugSymbolObjects4

2.2.1 IDebugClient

- 여러 인터페이스를 얻어오는 기능을 제공합니다

```
IDebugClient *          DebugClient;
PDEBUG_CONTROL    DebugControl;
HRESULT    Hr;

if ((Hr = DebugCreate(__uuidof(IDebugClient), (void **)&DebugClient)) != S_OK)
{
            return Hr;
}

if ((Hr = DebugClient ->QueryInterface(__uuidof(IDebugControl),
            (void **)&g_ExtControl)) != S_OK)
{
            goto Fail;
}

if ((Hr = DebugClient ->QueryInterface(__uuidof(IDebugSymbols2),
            (void **)&g_ExtSymbols)) != S_OK)
{
            goto Fail;
}
```

2.2.2 IDebugControl

- 디버깅을 제어하는 함수를 포함합니다(대표적인 함수만 보여줍니다)

함수	의미
OutputVaList	문자열을 지정된 포멧으로 정리한뒤 엔진 클라이언트의 Output Callback 으로 문자열을 전달합니다
OutputStackTrace	지정된 스택프레임 또는 현재 스택프레임을 Output Callback으로 전달합니다
Output	문자열을 엔진 클라이언트의 Output Callback 으로 문자열을 전달합니다
Execute	지정하는 디버거명령어를 실행합니다
GetStackTrace	지정하는 콜스택의 최상위 스택프레임을 리턴합니다
Evaluate	지정하는 문자열의 포멧을 조사합니다
GetWindbgExtensionApis64	WINDBG_EXTENSION_APIS64 구조체를 얻습니다
GetActualProcessorType	디버깅하는 타겟의 CPU 유형을 얻습니다
….	

2.2.3 IDebugSymbols

- 심볼과 관련된 작업을 돕습니다

함수	의미
GetSymbolTypeId	지정하는 심볼을 위한 TypeID를 구합니다
GetFieldName	지정하는 심볼(TypeID)과 위치에 해당하는 필드 이름을 구합니다
GetFieldOffset	지정하는 심볼(TypeID)과 위치에 해당하는 필드 오프셋 값을 구합니다
GetConstantName	지정하는 값에 대응 되는 상수이름을 얻습니다
GetScopeSymbolGroup	지정하는 타겟에 포함된 심볼그룹정보를 얻습니다
SetScope	제공하는 스택프레임, 명령실행위치, 레지스터문맥을 현재 문맥으로 지정합니다
….	

2.2.3.1 WDbgExts 가 제공하는 매크로함수

- 심볼에 대응되는 데이터를 파싱하는 작업을 돕기 위해서 WDbgExts.h 파일은 다양한 매크로함수를 제공하고 있습니다.
- 반드시 ExtensionApis 전역변수의 값을 초기화해야 사용할 수 있습니다.

[사용하려는 측]

```
ExtensionApis.nSize = sizeof(ExtensionApis);
Hr = DebugControl->GetWindbgExtensionApis64(&ExtensionApis);
```

[WDbgExts.h] 기본 매크로

```
#define dprintf                 (ExtensionApis.lpOutputRoutine)
#define GetExpression           (ExtensionApis.lpGetExpressionRoutine)
#define CheckControlC           (ExtensionApis.lpCheckControlCRoutine)
#define GetContext              (ExtensionApis.lpGetThreadContextRoutine)
#define SetContext              (ExtensionApis.lpSetThreadContextRoutine)
#define Ioctl           (ExtensionApis.lpIoctlRoutine)
#define Disasm                  (ExtensionApis.lpDisasmRoutine)
#define GetSymbol               (ExtensionApis.lpGetSymbolRoutine)
#define ReadMemory              (ExtensionApis.lpReadProcessMemoryRoutine)
#define WriteMemory             (ExtensionApis.lpWriteProcessMemoryRoutine)
#define StackTrace              (ExtensionApis.lpStackTraceRoutine)
```

- 기본 매크로이외에 보다 복잡한 매크로를 추가로 지원하고 있습니다

매크로 함수(주요)	의미
__inline VOID ReadPhysical(ULONG64 address, _Out_writes_bytes_to_(size, *sizer) PVOID buf, ULONG size, _Out_ PULONG sizer)	물리메모리의 내용을 읽습니다
__inline VOID WritePhysical(ULONG64 address, _In_reads_bytes_(size) PVOID buf, ULONG size, _Out_ PULONG sizew)	물리메모리의 내용을 기록합니다
__inline VOID ReadIoSpace(ULONG address, _Out_writes_bytes_(*size) PULONG data, _Inout_ PULONG size)	입출력포트를 읽습니다
__inline VOID WriteIoSpace(ULONG address, ULONG data, _Inout_ PULONG size)	입출력포트를 기록하니다
__inline ULONG ReadPointer(ULONG64 Address, PULONG64 Pointer)	지정하는 위치의 내용을 포인터변수에 담습니다
__inline ULONG WritePointer(ULONG64 Address, ULONG64 Pointer)	지정하는 위치의 내용으로 포인터변수의 값을 기록합니다
__inline ULONG GetTypeSize (IN LPCSTR Type)	심볼의 바이트 크기를 알려줍니다
__inline ULONG GetFieldData (_In_ ULONG64 TypeAddress, _In_ LPCSTR Type, _In_ LPCSTR Field, _In_ ULONG OutSize, _Out_writes_bytes_(OutSize) PVOID pOutValue)	메모리의 내용을 심볼에 맞춘뒤 지정하는 필드의 값을 얻습니다

- 메모리의 내용을 특정 심볼(구조체)에 연결하고 필드의 값을 가져오는 예재입니다

[정의된 심볼]

```
typedef struct _EXCEPTION_RECORD {
    NTSTATUS ExceptionCode;
    ULONG ExceptionFlags;
    struct _EXCEPTION_RECORD *ExceptionRecord;
    PVOID ExceptionAddress;
    ULONG NumberParameters;
    ULONG_PTR ExceptionInformation[EXCEPTION_MAXIMUM_PARAMETERS];
    } EXCEPTION_RECORD;
```

[심볼을 사용해서 필드의 값을 가져오는 방법]

```
 GetFieldValue(Address, "_EXCEPTION_RECORD", "ExceptionAddress", ExceptionAddress);
```

Address 는 심볼을 연결하려는 가상메모리주소입니다/

3. DbgEng Extension

- Debugger 가 사용할 수 있도록 확장기능을 제공하는 모듈을 의미합니다
- Debuggger 에서 !XXXX.YYYY 형태로 사용합니다

(XXXX 모듈이름, YYYY 명령어)

명령어	Command
.load	지정하는 DbgEng Extension DLL 을 로드합니다

```
1: kd> .load usbkd.dll
USB Debugger Extension Ver 3
Use !usbhelp for a list of extension comma
1: kd> !usbkd.usbhelp
USB Debugger Extension Ver 3
Use !usbhelp for a list of extension co

!usb2tree
This command displays topology of the USB2 device tree.

!usbhcdlist
Dumps a list of all USBPORT Host Controllers.

!usbhcdlistlogs
Dumps a brief list of all USBPORT Host Controller FDOs,
and the complete debug logs for all EHCI Host Controllers.

!usbhcdext <addr>
        <addr> = address of a USBPORT FDO or PDO Device Extension
Dumps details about a given USBPORT Host Controller FDO or Root Hub PDO.

!ehci_info_from_fdo <addr>
        <addr> = address of a USBPORT Host Controller FDO Device Object
Dumps details about a given USBPORT Host Controller FDO.

!usbhcdlog <addr>[, <num_entries>]
        <addr> = address of a USBPORT Host Controller FDO Device Extension
        <num_entries> = number of entries to dump or -1 for all entries
Dumps a USBPORT Host Controller FDO debug log.

!usbhcdlogex <addr>[, <num_entries>]
        <addr> = address of a USBPORT Host Controller FDO Device Extension
        <num_entries> = number of entries to dump or -1 for all entries
Dumps an annotated USBPORT Host Controller FDO debug log.
```

Usbkd 모듈을 로드합니다

Usbkd 모듈이 제공하는 명령어(usbhelp)를 사용합니다

WinDBG 프로그램

Command

```
1: kd> .load usbkd.dll
USB Debugger Extension Ver 3
Use !usbhelp for a list of extension commands
1: kd> !usbkd.usbhelp
USB Debugger Extension Ver 3
Use !usbhelp for a list of extension commands

!usb2tree
This command displays topology of the USB2 device tree.

!usbhcdlist
Dumps a list of all USBPORT Host Controllers.

!usbhcdlistlogs
Dumps a brief list of all USBPORT Host Controller FDOs,
and the complete debug logs for all EHCI Host Controllers.

!usbhcdext <addr>
      <addr> = address of a USBPORT FDO or PDO Device Extension
Dumps details about a given USBPORT Host Controller FDO or Root Hub PDO.

!ehci_info_from_fdo <addr>
      <addr> = address of a USBPORT Host Controller FDO Device Object
Dumps details about a given USBPORT Host Controller FDO.

!usbhcdlog <addr>[, <num_entries>]
      <addr> = address of a USBPORT Host Controller FDO Device Extension
      <num_entries> = number of entries to dump or -1 for all entries
Dumps a USBPORT Host Controller FDO debug log.

!usbhcdlogex <addr>[, <num_entries>]
      <addr> = address of a USBPORT Host Controller FDO Device Extension
      <num_entries> = number of entries to dump or -1 for all entries
Dumps an annotated USBPORT Host Controller FDO debug log.

!usbhcdpow <addr>
      <addr> = address of a USBPORT FDO or PDO Device Extension
Dumps the power state history for the given USBPORT Host Controller FDO or Root Hub PDO
1: kd>
```

DbgEng Extension

예시

Usbkd.dll

출력

입력

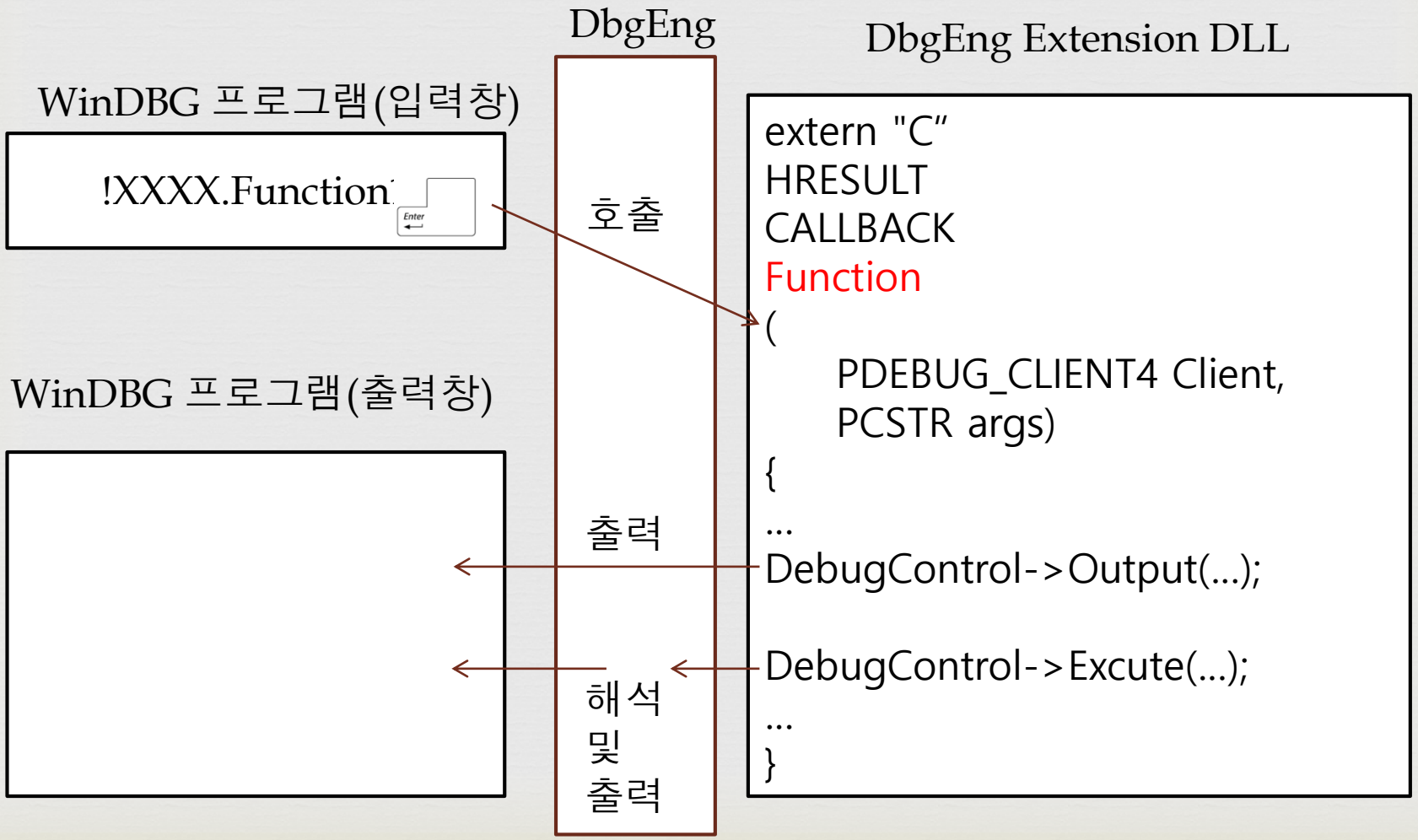
3.1 필수 Export 함수

- DbgEng Extension 이 제공하는 필수 Export 함수를 다음과 같습니다

Export 함수	의미
extern "C" HRESULT CALLBACK DebugExtensionInitialize(PULONG Version, PULONG Flags)	메모리에 로드될 때 호출됩니다
extern "C" void CALLBACK DebugExtensionNotify(ULONG Notify, ULONG64 Argument)	정해진 사건들이 발생될때 호출됩니다
extern "C" void CALLBACK DebugExtensionUninitialize(void)	메모리에서 해제될 때 호출됩니다
extern "C" HRESULT CALLBACK KnownStructOutput(__in ULONG Flag, __in ULONG64 Address, __in PSTR StructName, __out_ecount(BufferSize) PSTR Buffer, __in PULONG BufferSize)	해석할 수 있는 심볼(스트럭쳐)정보를 제공하고 해석합니다
extern "C" HRESULT _EFN_Analyze(__in PDEBUG_CLIENT4 Client, __in FA_EXTENSION_PLUGIN_PHASE CallPhase, __in PDEBUG_FAILURE_ANALYSIS2 pAnalysis)	덤프분석용도로 사용되는 특별한 함수입니다 !Analyze 명령어와 함께 사용됩니다

3.2 선택 Export 함수(명령어)

- DbgEng Extension 이 제공하는 선택 Export 함수는 명령어입니다



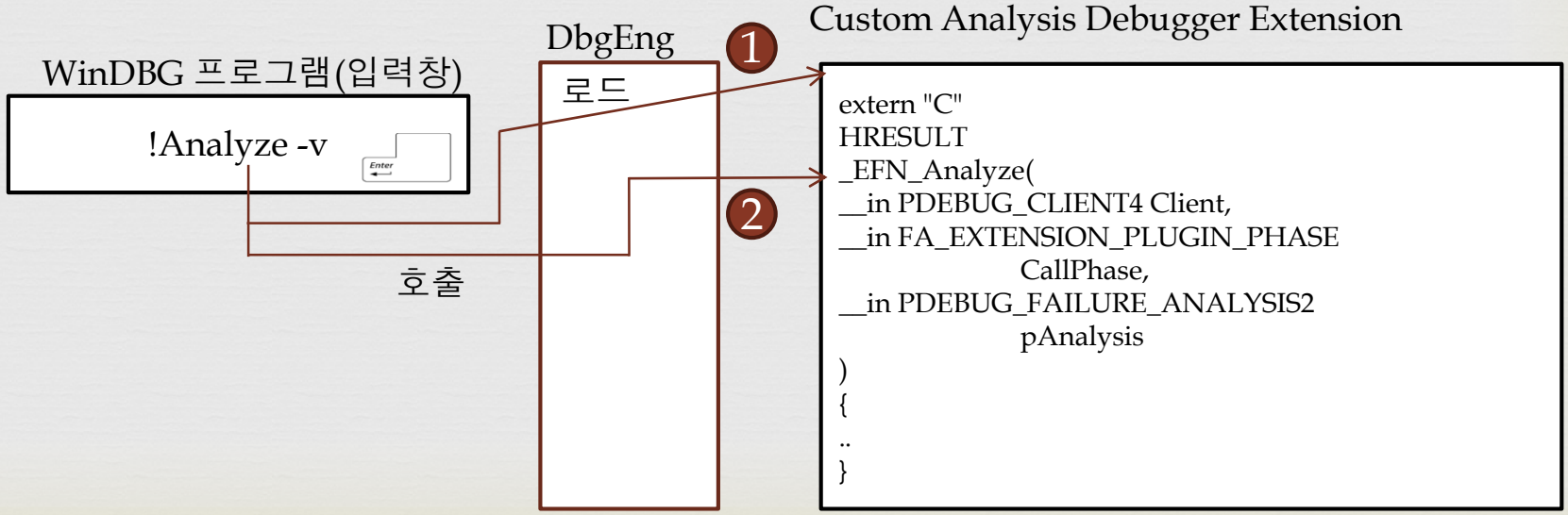
3.3 Custom Analysis Debugger Extension

- WinDBG 가 사용하는 !Analyze 명령어를 도와주는 확장모듈입니다
- _EFN_Analyze 함수를 Export 해야 합니다
- Metafile 을 제공해야 합니다. 이 파일은 Extension DLL 과 같은 파일이름을 사용하고 확장자를 'ALZ' 를 사용합니다

MyPlugin.ALZ

PluginId	MyPlugin
DebuggeeClass	Kernel
BugCheckCode	0xA
BugCheckCode	0xE2

분석을 지원하는 BugCheck 코드를 기록합니다



3.3.1 예시코드

```
#include <windows.h>
#define KDEXT_64BIT // 64비트환경을 지원하는 경우에 사용합니다
#include <wdbgexts.h>
#include <dbgeng.h>
#include <extsfns.h>

extern "C" __declspec(dllexport) HRESULT _EFN_Analyze(_In_ PDEBUG_CLIENT4 Client,
                _In_ FA_EXTENSION_PLUGIN_PHASE CallPhase,
                _In_ PDEBUG_FAILURE_ANALYSIS2 pAnalysis)
{
   HRESULT hr = E_FAIL;
   PDEBUG_CONTROL pControl = NULL;
   hr = Client->QueryInterface(__uuidof(IDebugControl), (void**)&pControl);
   if(S_OK == hr && NULL != pControl)
   {
      IDebugFAEntryTags* pTags = NULL;
      pAnalysis->GetDebugFATagControl(&pTags);
      if(NULL != pTags)
      {
         if(FA_PLUGIN_INITILIZATION == CallPhase)
         {
            pControl->Output(DEBUG_OUTPUT_NORMAL, "My analyzer: initialization₩n");
         }
```

```
extern "C" __declspec(dllexport) HRESULT _EFN_Analyze(_In_ PDEBUG_CLIENT4 Client,
                _In_ FA_EXTENSION_PLUGIN_PHASE CallPhase,
                _In_ PDEBUG_FAILURE_ANALYSIS2 pAnalysis)
{
    ...
        else if(FA_PLUGIN_STACK_ANALYSIS == CallPhase)
        {
           pControl->Output(DEBUG_OUTPUT_NORMAL, "My analyzer: stack analysis₩n");
        }
        else if(FA_PLUGIN_PRE_BUCKETING == CallPhase)
        {
           pControl->Output(DEBUG_OUTPUT_NORMAL, "My analyzer: prebucketing₩n");
        }
        else if(FA_PLUGIN_POST_BUCKETING == CallPhase)
        {
           pControl->Output(DEBUG_OUTPUT_NORMAL, "My analyzer: post bucketing₩n");
           FA_ENTRY_TYPE entryType = pTags->GetType(DEBUG_FLR_BUGCHECK_CODE);
           pControl->Output(DEBUG_OUTPUT_NORMAL, "The data type for the
                           DEBUG_FLR_BUGCHECK_CODE tag is 0x%x.₩n₩n", entryType);
        }
     }

     pControl->Release();
  }
  return hr;
}
```

3.3.2 출력

```
*                        Bugcheck Analysis                        *
*                                                                 *
*******************************************************************************

Use !analyze -v to get detailed debugging information.

BugCheck E2, {0, 0, 0, 0}

My analyzer: initialization
My analyzer: stack analysis
My analyzer: prebucketing
My analyzer: post bucketing
The data type for the DEBUG_FLR_BUGCHECK_CODE tag is 0x1.
```

4. DbgEng Extension 예재

- 간단한 기능을 가지는 DbgEng Extension DLL 모듈을 만들어봅니다
- 4개의 명령어를 지원합니다

Export 함수	의미
HRESULT CALLBACK cmdsample (PDEBUG_CLIENT4 Client, PCSTR args)	lm 명령을 사용하고, 사용자의 입력을 받아서 실행합니다
HRESULT CALLBACK structsample (PDEBUG_CLIENT4 Client, PCSTR args)	주어진 메모리를 스트럭쳐에 입혀서 출력합니다
HRESULT CALLBACK help (PDEBUG_CLIENT4 Client, PCSTR args)	명령사용방법을 보여줍니다

- WinDBG가 설치된 경로아래에 Extension 모듈 경로로 복사해야 합니다

"C:\Program Files (x86)\Windows Kits\10\Debuggers\x86\winext"

4.1 출력

```
1: kd> !dbgexts.help
Help for dbgexts.dll
  cmdsample         - This does stacktrace and lists
  help              = Shows this help
  structsample <addr> - This dumps a struct at given address
1: kd> !dbgexts.structsample c0000000
c0000000: 00000000 00000000 00000000 00000000

Method 1:
_EXCEPTION_RECORD @ c0000000
                ExceptionCode         : 0
                ExceptionAddress      : 00000000
                ExceptionInformation[1] : 0


Method 2:
_EXCEPTION_RECORD @ c0000000
                ExceptionCode         : 0
                ExceptionAddress      : 00000000
                ExceptionInformation[1] : 0
Fields of _EXCEPTION_RECORD
0 (+000) ExceptionCode
1 (+004) ExceptionFlags
2 (+008) ExceptionRecord
3 (+00c) ExceptionAddress
4 (+010) NumberParameters
5 (+014) ExceptionInformation
Testenum 0 == _EXCEPTION_RECORD
```

임의의 주소 0xC0000000 을 사용했습니다

```
typedef struct _EXCEPTION_RECORD {
  NTSTATUS ExceptionCode;
  ULONG ExceptionFlags;
  struct _EXCEPTION_RECORD *ExceptionRecord;
  PVOID ExceptionAddress;
  ULONG NumberParameters;
  ULONG_PTR
ExceptionInformation[EXCEPTION_MAXIMUM_PARAMETERS];
  } EXCEPTION_RECORD;
```

4.2 샘플소스

- DbgEng Extension 샘플코드는 하제소프트 홈페이지에서 받으실 수 있습니다(마이크로 소프트가 제공하는 예재를 사용했습니다)
- 샘플코드는 Visual Studio 2015 에서 빌드할 수 있도록 수정하였습니다

수고하셨습니다

주식회사 하제소프트

(www.hajesoft.co.kr)

강사 이 봉 석