請以純粹黑白列印

更有效率的使用 **DLL**

Delay-Loading a DLL

井民全製作

當你的程式使用的 DLL 越來越多時,
你會發現你的程式載入時間越來越久

另外, 如果你有些 DLL 並不支援舊的作業系統.
你會希望先讓程式執行, 然後根據情況載入適當的 DLL

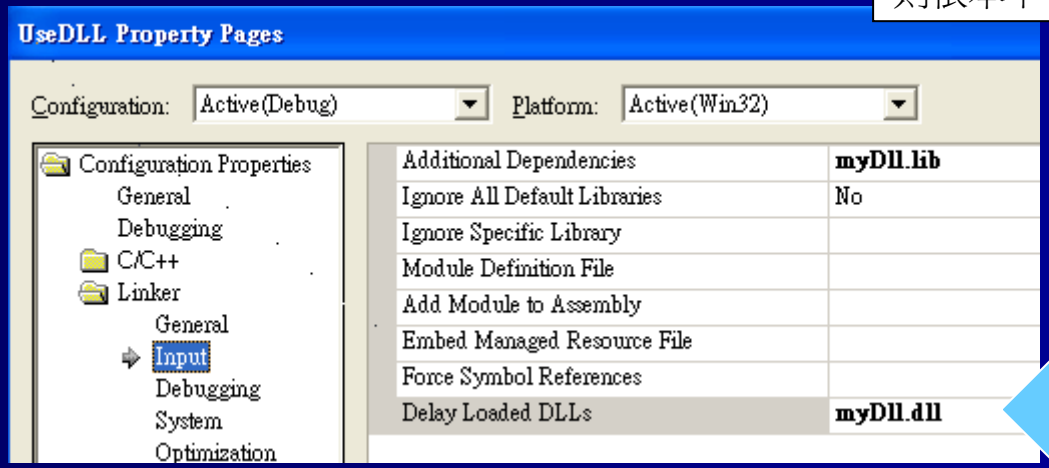
- 你可以使用 Delay-loading 解決你的問題
  - 可以在存取該 DLL export 的 symbol 時才載入該 DLL
  - Delay-Loading 是一些小程式, 幫你自動 Loadlibrary

放在 DELAYIMP.LIB

好了,請告訴我如何使用 DelayLoad

- 按照一般方式建立 DLL
- 按照一般方式建立 執行檔

若你沒有使用該 function,
則根本不會載入對應的 DLL



當 DLL 被載入時,會呼叫 DllMain, 你可以藉此驗證

加入 /DelayLoad:MyDll.dll 會自動加入Delayimp.lib

呼叫 Dll export symbol 的流程

```
#include "MyLib.h"          // << ---------- 引入必要的標頭檔
#include <iostream>
using namespace std;

int _tmain( ) {
    int d=Add(1,2);
    cout << d << endl;
    return 0;
}
```

Linker 幫你將 function 的呼叫轉換成呼叫 __delayLoadHelper

呼叫 __delayLoadHelper

當第一次呼叫 function 時，就會自動呼叫 LoadLibrary，幫你載入需要的 DLL

檢查 DLL 是否已經載入

呼叫 LoadLibrary 載入 DLL

執行 DllMain

呼叫 GetProcAddress 取得 Function 位址

有可能會發生的問題

因爲是呼叫的時候，才載入對應的 DLL 故可能會發生

- __delayLoadHelper 找不到對應的 DLL
  - Exception 發生
- __delayLoadHelper 找到DLL 但是卻沒有你呼叫的 function
  - Exception 發生

例如舊版的 DLL

若 DLL 發生了問題,詳細情況如何?

```
#include <Delayimp.h>

typedef struct DelayLoadInfo {
    DWORD                   cb;         // Size of structure
    PCImgDelayDescr         pidd;       // Raw data (everything is there)
    FARPROC *               ppfn;       // Points to address of function to load
    LPCSTR                  szDll;      // 你載入的DLL 名稱
    DelayLoadProc           dlp;        // 紀錄你呼叫的
    HMODULE                 hmodCur;    // hInstance of loaded library
    FARPROC                 pfnCur;     // Actual function that will be called
    DWORD                   dwLastError;// Error received
} DelayLoadInfo, * PDelayLoadInfo;
```

DLL Name

Procedure Name

```
typedef struct DelayLoadProc {
    BOOL fImportByName;
    union {
        LPCSTR szProcName;
        DWORD  dwOrdinal;
    };
} DelayLoadProc;
```

判斷是否用Name呼叫

Procedure Name

序號方式呼叫

好了,告訴我怎麼抓 Exception

```
#include <Delayimp.h>
int _tmain(int argc, _TCHAR* argv[]) {
// Step 1: 嘗試載入 Add function from MyDll.dll
    __try {
            int d=Add(1,2);
            cout << d << endl;

// Step 2: 攔截發生的 Exception
            }__except (DelayLoadDllExceptionFilter(GetExceptionInformation( ))) {
                    // Nothing to do in here, thread continues to run normally
            }
            return 0;
}
```

取得 Exception 狀態描述

呼叫我們自己寫的判斷函式

試著把 **MyDLL.dll** 拿掉, 讓系統找不到看看 !!

```
LONG WINAPI DelayLoadDllExceptionFilter(PEXCEPTION_POINTERS pep) {

    PDelayLoadInfo pdli;

    pdli =  PDelayLoadInfo( pep->ExceptionRecord->ExceptionInformation[0]);

    char sz[500] = { 0 };// Error Message 使用

    // Step 2: 判斷目前的 Exception 種類
    switch (pep->ExceptionRecord->ExceptionCode) {
        case VcppException(ERROR_SEVERITY_ERROR, ERROR_MOD_NOT_FOUND):
            wsprintfA(sz, "DLL module 沒有找到: %s", pdli->szDll);
            break;

        case VcppException(ERROR_SEVERITY_ERROR, ERROR_PROC_NOT_FOUND):
                wsprintfA(sz, "找到 DLL module %s, 但是找不到呼叫的 function %s",
                                                    pdli->szDll, pdli->dlp.szProcName);
                break;
    }
}
```

Step 1: 取得 Exception 資訊並
將資料填入 pDelayLoadInformation 結構

前面定義的 Delay Load 資料結構

DLL Module Not Found

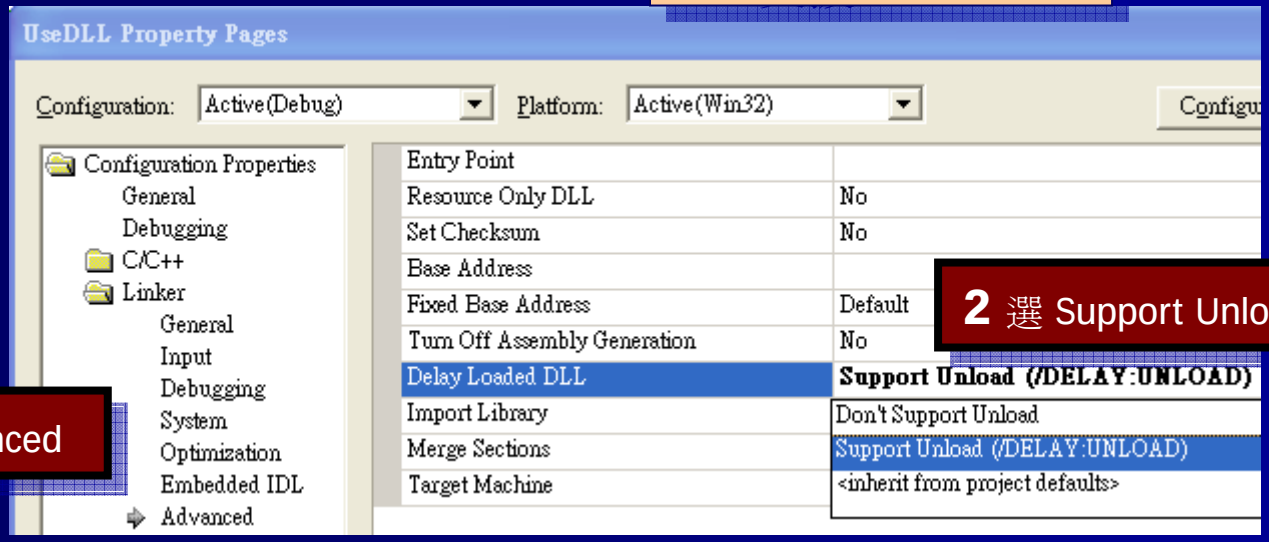
Procedure Not Found

詳細程式請看範例

節省記憶體的使用量

當我不想再用該 **DLL** 中所有 **function**時, 我可不可以 **Upload**

VC 支援 unload DLL



1 選 Advanced

2 選 Support Unload

請看範例

```
#include <windows.h>
#include <delayimp.h>          // for __FUnloadDelayLoadedDLL2

#include "MyLib.h"
#include <iostream>
using namespace std;

int _tmain(int argc, _TCHAR* argv[]) {
// MyDLL.DLL will load at this point
   int d=Add(1,2);
   cout << d << endl;

// MyDLL.DLL will unload at this point
   BOOL TestReturn = __FUnloadDelayLoadedDLL2("MyDLL.dll");
   if (TestReturn)
        printf("\nDLL was unloaded");
   else
        printf("\nDLL was not unloaded");
   return 0;
}
```

注意:
你必須在專案中使用 **DelayLoad** 否則會找不到 **_FUnloadDelayLoadedDLL2** 這個 **function**

Unload 你的 DLL

注意:
1. 不能包含 **Path Name**
2. 大小寫必須與 **Delayload**的設定一樣
3. 請確定你沒有自行呼叫 **FreeLibrary**

更正

有用的 function

- 判斷 module 是否已經載入記憶體中

```
void main(){
    IsModuleLoaded("MyDLL");
}
```

IsLoadedCheck

```
void IsModuleLoaded(PCTSTR pszModuleName) {
    HMODULE hmod = GetModuleHandle(pszModuleName);
    char sz[100];
#ifdef UNICODE
    wsprintfA(sz, "Module \"%S\" is %Sloaded.",
    pszModuleName, (hmod == NULL) ? L"not " : L"");
#else
    wsprintfA(sz, "Module \"%s\" is %sloaded.",
    pszModuleName, (hmod == NULL) ? "not " : "");
#endif
    MessageBox(NULL,sz,"Information",MB_OK);
}
```

若傳回 **NULL** 表示尚未載入記憶體

進階話題:
掌控 __delayLoadHelper 的處理進度

更進一步的想攔截
delayLoad 的功能

你可以指定下面兩個
function 指標

處理進行過程

PfnDliHook __pfnDliNotifyHook2= DliHook;
PfnDliHook __**pfnDliFailureHook2**= DliHook;

錯誤處理過程

你的 callback function

進階話題:
掌控 __delayLoadHelper 的處理進度

- 在DelayImp.lib這個靜態連結函式庫中，定義了兩個全域變數
  - __pfnDliNotifyHook
  - __pfnDliFailureHook

其實是 **function pointer**

一看就知道是
Function pointer

變數的型態

```
typedef FARPROC (WINAPI *PfnDliHook)(
                                    unsigned dliNotify,
                                    PDelayLoadInfo pdli);
```

```
PfnDliHook __pfnDliNotifyHook = DliHook;
PfnDliHook __pfnDliFailureHook = DliHook;
```

註冊你的 **function**

看一下範例

```
#include <windows.h>
#include <tchar.h>
#include <delayimp.h>          // for PfnDliHook

#include "MyLib.h"

// 你的 call back function
FARPROC WINAPI DliHook(unsigned dliNotify, PDelayLoadInfo pdli);

// Tell __delayLoadHelper to call my hook function
PfnDliHook __pfnDliNotifyHook2  = DliHook;
PfnDliHook __pfnDliFailureHook2 = DliHook;

int _tmain(int argc, TCHAR* argv[]){
    int d=Add(1,2);
   __FUnloadDelayLoadedDLL2("MyDLL.dll");
}
```

1

DliHook function 在下一頁

2

指定你的
callback function

FARPROC WINAPI DliHook (unsigned dliNotify, PDelayLoadInfo pdli)

dliStartProcessing
當 Helper 找尋適當的 DLL 時

dliNotePreLoadLibrary
當 Helper 呼叫 LoadLibrary 載入你的 DLL 之前

dliFailLoadLib
載入 DLL 失敗時

dliNotePreGetProcAddress
取得 function 位址之前

dliFailGetProc
取 function 位址失敗時

dliNoteEndProcessing
當 Helper 完成工作時

Switch - Case

```
FARPROC WINAPI DliHook(unsigned dliNotify, PDelayLoadInfo pdli) {

  FARPROC fp = NULL;   // 預設傳回值

  switch (dliNotify) {
  case dliStartProcessing:
            MessageBox(NULL,"企圖尋找一個 DLL
    break;

  case dliNotePreLoadLibrary:
      fp = (FARPROC) (HMODULE) NULL;
      MessageBox(NULL,"before LoadLibrary","Info",MB_
    break;

  case dliFailLoadLib:
            fp = (FARPROC) (HMODULE) NULL;
             MessageBox(NULL," LoadLibrary fails","Info",MB_OK);
    break;
  case dliNotePreGetProcAddress:
            fp = (FARPROC) NULL;
             MessageBox(NULL,"before GetProcAddress","Info",MB_OK);
    break;
```

尋找一個 DLL function
傳回值:
NULL ➔ 不做事 ;
nonzero ➔ override everything

載入你的 DLL 之前
傳回值:
NULL ➔ 系統會自行呼叫 LoadLibrary
HMODULE ➔你自行呼叫 LoadLibrary

載入 DLL 失敗時
傳回值:
NULL ➔會發出 ERROR_MOD_NOT_FOUND 例外
HMODULE ➔自行處理載入的動作,並傳回載入的 HMODULE

取得 function 位址之前
傳回值:
NULL ➔ 系統會自行呼叫 GetProcAddress
FARPROC ➔ 你自行呼叫 GetProcAddress,並傳回 function 的位址

```
case dliFailGetProc:
        fp = (FARPROC) NULL;
        MessageBox(NULL,"GetProcAddress fails","Info",MB_OK);
    break;
```

取 function 位址失敗時
傳回值:
NULL :
　會發出 ERROR_PROC_NOT_FOUND 例外
FARPROC:
　自行處理取出位址的更正的動作,並傳回載入的 FARPROC

```
  case dliNoteEndProcessing:
    // 當 Helper 完成工作時
    // 你可以簡單的取出 pdli 結構中的資料或發出 Exception, 隨便你
        MessageBox(NULL,"done","Info",MB_OK);
    break;
  }

  return(fp);
}
```

詳細程式請看範例
DelayLoadHook

- End