

```

/*****
/*
/*----- P E I M E X . C -----*/
/* Task      : Displays the import and export sections of a file */
/*            (in vitro) */
/*-----*/
/* Authors    : Michael Tischer and Bruno Jennrich */
/* developed on : 09/04/1995 */
/* last update  : 09/06/1995 */
/*****
#include <windows.h>
#include <stdio.h>
#include <conio.h>

/*****
/* main : Here we go */
/*-----*/
/* Parameters: argc - Argument counter */
/*            argv - Argument values */
/*            argv[1] - Name of EXE/DLL file to be examined */
/*****
void main( int argc, char *argv[] )
{
    HANDLE hFile; // Handle to EXE/DLL file
    HANDLE hFileMapping; // Handle to memory map

    hFile = CreateFile( argv[ 1 ], // Open file reading
                        GENERIC_READ,
                        FILE_SHARE_READ,
                        NULL,
                        OPEN_EXISTING,
                        0,
                        NULL );
    if( hFile != INVALID_HANDLE_VALUE )
    {
        // Create memory map file for file -----
        hFileMapping = CreateFileMapping( hFile,
                                         NULL,
                                         PAGE_READONLY | SEC_COMMIT,
                                         0,
                                         0,
                                         NULL );

        if( hFileMapping )
        {
            // Create map of memory map file -----
            LPBYTE pByte = MapViewOfFile( hFileMapping,
                                         FILE_MAP_READ,
                                         0,
                                         0,
                                         0 );

            if( pByte )
            {
                PIMAGE_DOS_HEADER pDH;
                PIMAGE_NT_HEADERS pNTH;
                PIMAGE_SECTION_HEADER pSH;
                PIMAGE_IMPORT_DESCRIPTOR pIID;
                int i;

                // There is a DosHeader at the BaseAddress -----
                pDH = ( PIMAGE_DOS_HEADER ) pByte;

                // Is it really a DosHeader ( MZ )? -----
                if( pDH->e_magic == IMAGE_DOS_SIGNATURE )
                {
                    // The DosHeader is followed by an NTHeader -----
                    pNTH = ( PIMAGE_NT_HEADERS )
                        ( ( DWORD ) pByte +
                          ( DWORD ) pDH->e_lfanew );
                    // Is it really an NTHeader ( PE )? -----
                    if( pNTH->Signature == IMAGE_NT_SIGNATURE ) // PE
                    {
                        // After the NTHeader come the SectionHeaders -----
                        pSH = ( PIMAGE_SECTION_HEADER ) ( ( DWORD ) pNTH +
                                                          sizeof( IMAGE_NT_HEADERS ) );
                    }
                }
            }
        }
    }
}

```

```

// Browse all SectionHeaders and search for -----
// .idata and .edata -----
for( i = 0;
    i < pNTH->FileHeader.NumberOfSections;
    i++ )
{
    if( strcmp( pSH[ i ].Name, ".idata" ) == 0 )
    {
        // .idata found!
        int j;
        int delta; // Correction value

        // VirtualAddress - Address of the data when loaded -
        //                      as an image by the Program Loader -
        // PointerToRawData - Address within the file -
        delta = (pSH[ i ].VirtualAddress -
                pSH[ i ].PointerToRawData );

        j = 0;
        do
        {
            // In the .idata-Section you will first find the --
            // IMAGE_IMPORT_DESCRIPTOR, which exist for each --
            // DLL from which functions are imported --
            pIID = ( PIMAGE_IMPORT_DESCRIPTOR )
                ( ( DWORD ) pDH +
                  ( DWORD ) pSH[i].PointerToRawData +
                    j * sizeof( IMAGE_IMPORT_DESCRIPTOR ) );

            // Valid IMAGE_IMPORT_DESCRIPTOR? -----
            if( pIID->Name )
            {
                LPSTR          pDLLName;
                PIMAGE_THUNK_DATA pThunk;
                PIMAGE_IMPORT_BY_NAME pIIBN;

                // Get address of DLLName -----
                pDLLName = ( LPSTR )
                    ( ( DWORD ) pDH +
                      ( DWORD ) pIID->Name - delta );

                // Get address of function descriptor -----
                pThunk = ( PIMAGE_THUNK_DATA )
                    ( ( DWORD ) pDH +
                      ( DWORD ) pIID->Characteristics - delta );

                // Does descriptor point to function? -----
                while( pThunk->ul.AddressOfData )
                {
                    // Only ordinal number of function known?
                    if( ( pThunk->ul.Ordinal & IMAGE_ORDINAL_FLAG ) )
                        printf( "%s @%ld\n",
                                pDLLName,
                                pThunk->ul.Ordinal & 0xFFFF );
                    else
                    {
                        pIIBN = ( PIMAGE_IMPORT_BY_NAME )
                            ( ( DWORD ) pDH +
                              ( DWORD ) pThunk->ul.AddressOfData -
                                delta );

                        printf( "%s %s\n",
                                pDLLName,
                                pIIBN->Name );
                    }
                    pThunk++;
                }
                j++;
            }
            while ( pIID->Name );
        }
        else
        if( strcmp( pSH[ i ].Name, ".edata" ) == 0 )
        {
            int delta; // .edata found !
            DWORD j;

```

```

PIMAGE_EXPORT_DIRECTORY pExports;
LPSTR                    pFileName;
PDWORD                  pFunctions; // Function addresses
PWORD                   pOrdinals;  // Ordinal numbers
PSTR                    *pNames;    // Function names

// We also need a correction value here -----
delta = (pSH[ i ].VirtualAddress -
        pSH[ i ].PointerToRawData );

pExports = ( PIMAGE_EXPORT_DIRECTORY )
            ( ( DWORD )pDH +
              ( DWORD )pSH[ i ].PointerToRawData );

pFileName = (PSTR)( ( DWORD )pDH +
                   ( DWORD )pExports->Name - delta );

printf( "%s\n", pFileName );

// Get address for array with function addresses
pFunctions = ( PDWORD )
              ( ( DWORD )pDH +
                ( DWORD )pExports->AddressOfFunctions
                - delta );

// Get address for array with ordinal numbers -----
pOrdinals = ( PWORD )
             ( ( DWORD )pDH +
               ( DWORD )pExports->AddressOfNameOrdinals
               - delta );

// Get address for array with addresses of names
pNames = (PSTR *) ( ( DWORD )pDH +
                   ( DWORD )pExports->AddressOfNames
                   - delta );

// Browse all exported functions -----
for ( j=0; j < pExports->NumberOfNames; j++ )
{
    printf( " %08X %4u %s\n",
            *pFunctions,
            *pOrdinals,
            ( ( DWORD ) pDH + ( DWORD ) *pNames - delta ));
    pNames++;
    pOrdinals++;
    pFunctions++;
}
}
}
printf("<Key>");
_getch();
}
else printf("Not PE-Header");
}
else printf("Not DOS-Header");
UnmapViewOfFile( pByte );
}
CloseHandle( hFileMapping );
}
else printf("Cannot open memory map file\n");
CloseHandle( hFile );
}
else printf("Cannot open file\n");
}

```