Microsoft's PE Checksum Algorithm / microsoft_pe_checksum_algo_distilled.c

```
//
// The following code was taken from the publicly released IMAGEHLP SDK tools code from
// the Windows NT 3.51 Win32 SDK (June 1995) distribution. For convenience and clarity, I
// made some superficial changes to the code shown here:
//
//  -I included [within this file] only the code portions that pertain to the PE checksum
//   algorithm, associated APIs and other private functions referenced
//  -some of the more liberal whitespace has been removed (e.g. multiline function arguments
//   lists, multiline comments collapsed into single line, refactored whitespace and position
//   of content within function comment headers such as the author info, indentation and
//   position of some of the blocks of code and individual statements have been altered
//   for formatting consistency, etc.)
//  -the code statements themselves are otherwise in their original form
//  -if you want to compile this file directly, you'll need to make some small changes,
//   the least of which would be to include "windows.h"
//
// This file begins with Microsoft's original copyright header from CHECKSUM.C:

/*++

Copyright (c) 1993  Microsoft Corporation

Module Name:

    checksum.c

Abstract:

    This module implements a function for computing the checksum of an
    image file. It will also compute the checksum of other files as well.

Author:

    David N. Cutler (davec) 21-Mar-1993

Revision History:

--*/

//
// ImageNtHeader() - This function returns the address of the NT Header.
//
// ARGUMENTS
//
//  Base            Supplies the base of the image.
//
// RETURN VALUE     Returns the address of the NT Header.
//
// FROM "imagehlp/imagedir.c": Steve Wood 18-Aug-1989
//
PIMAGE_NT_HEADERS ImageNtHeader(IN PVOID Base)
{
    PIMAGE_NT_HEADERS NtHeaders;
    if (Base != NULL && Base != (PVOID)-1)
    {
        try
        {
            if (((PIMAGE_DOS_HEADER)Base)->e_magic == IMAGE_DOS_SIGNATURE)
            {
                NtHeaders = (PIMAGE_NT_HEADERS)((PCHAR)Base + ((PIMAGE_DOS_HEADER)Base)->e_lfanew);
                if (NtHeaders->Signature == IMAGE_NT_SIGNATURE)
                {
                    return NtHeaders;
                }
            }
        }
        except(EXCEPTION_EXECUTE_HANDLER)
        {
            return NULL;
        }
    }
    return NULL;

} //ImageNtHeader()

//
// ChkSum() - Compute a partial checksum on a portion of an imagefile.
//
// ARGUMENTS
//
//  PartialSum      Supplies the initial checksum value.
//  Sources         Supplies a pointer to the array of words for which the checksum is computed.
//  Length          Supplies the length of the array in words.
//
// RETURN VALUE     The computed checksum value is returned as the function value.
//
// FROM "imagehlp/imagedir.c": Steve Wood 18-Aug-1989
//
USHORT ChkSum(ULONG PartialSum, PUSHORT Source, ULONG Length)
{
    // Compute the word wise checksum allowing carries to occur into the
    // high order half of the checksum longword.
    while (Length--)
    {
        PartialSum += *Source++;
        PartialSum = (PartialSum >> 16) + (PartialSum & 0xffff);
    }

    //
    // Fold final carry into a single word result and return the resultant
    // value.
    //
    return (USHORT)(((PartialSum >> 16) + PartialSum) & 0xffff);
```

```
} //ChkSum()

//
// CheckSumMappedFile() - This functions computes the checksum of a mapped file.
//
// ARGUMENTS
//
//  BaseAddress     Supplies a pointer to the base of the mapped file.
//  FileLength      Supplies the length of the file in bytes.
//  HeaderSum       Suppllies a pointer to a variable that receives the checksum from the image
//                  file, or zero if the file is not an image file.
//  CheckSum        Supplies a pointer to the variable that receive the computed checksum.
//
// RETURN VALUE     None
//
// FROM "imagehlp/checksum.c": David N. Cutler 21-Mar-1993
//
PIMAGE_NT_HEADERS CheckSumMappedFile(LPVOID BaseAddress, DWORD FileLength, LPDWORD HeaderSum, LPDWORD CheckSum)
{
    PUSHORT AdjustSum;
    PIMAGE_NT_HEADERS NtHeaders;
    USHORT PartialSum;

    //Compute the checksum of the file and zero the header checksum value.
    *HeaderSum = 0;
    PartialSum = ChkSum(0,(PUSHORT)BaseAddress,(FileLength + 1) >> 1);

    // If the file is an image file, then subtract the two checksum words
    // in the optional header from the computed checksum before adding
    // the file length, and set the value of the header checksum.
    try
    {
        NtHeaders = ImageNtHeader(BaseAddress);
    }
    except(EXCEPTION_EXECUTE_HANDLER)
    {
        NtHeaders = NULL;
    }

    if ((NtHeaders != NULL) && (NtHeaders != BaseAddress))
    {
        *HeaderSum = NtHeaders->OptionalHeader.CheckSum;
        AdjustSum = (PUSHORT)(&NtHeaders->OptionalHeader.CheckSum);
        PartialSum -= (PartialSum < AdjustSum[0]);
        PartialSum -= AdjustSum[0];
        PartialSum -= (PartialSum < AdjustSum[1]);
        PartialSum -= AdjustSum[1];
    }

    // Compute the final checksum value as the sum of the paritial checksum
    // and the file length.
    *CheckSum = (DWORD)PartialSum + FileLength;
    return NtHeaders;

} //CheckSumMappedFile()

//
// MapFileAndCheckSumW() - This functions maps the specified file and computes the checksum of the file.
//
// ARGUMENTS
//  Filename        Supplies a pointer to the name of the file whose checksum is computed.
//  HeaderSum       Supplies a pointer to a variable that receives the checksum from the image file,
//                  or zero if the file is not an image file.
//  CheckSum        Supplies a pointer to the variable that receive the computed checksum.
//
// RETURN VALUE     0 if successful, else error number.
//
// FROM "imagehlp/checksum.c": David N. Cutler 21-Mar-1993
//
DWORD MapFileAndCheckSumW(PWSTR Filename,LPDWORD HeaderSum,LPDWORD CheckSum)
{
    HANDLE FileHandle, MappingHandle;
    LPVOID BaseAddress;
    DWORD FileLength;

    // Open the file for read access
    FileHandle = CreateFileW(Filename,GENERIC_READ,FILE_SHARE_READ|FILE_SHARE_WRITE,NULL,OPEN_EXISTING,FILE_ATTRIBUTE_NORMAL,NULL);
    if (FileHandle == INVALID_HANDLE_VALUE)
    {
        return CHECKSUM_OPEN_FAILURE;
    }

    // Create a file mapping, map a view of the file into memory,
    // and close the file mapping handle.
    MappingHandle = CreateFileMapping(FileHandle,NULL,PAGE_READONLY,0,0,NULL);
    if (!MappingHandle)
    {
        CloseHandle(FileHandle);
        return CHECKSUM_MAP_FAILURE;
    }

    // Map a view of the file
    BaseAddress = MapViewOfFile(MappingHandle,FILE_MAP_READ,0,0,0);
    CloseHandle(MappingHandle);
    if (BaseAddress == NULL)
    {
        CloseHandle(FileHandle);
        return CHECKSUM_MAPVIEW_FAILURE;
    }

    // Get the length of the file in bytes and compute the checksum.
    FileLength = GetFileSize(FileHandle,NULL);
    CheckSumMappedFile(BaseAddress,FileLength,HeaderSum,CheckSum);

    // Unmap the view of the file and close file handle.
    UnmapViewOfFile(BaseAddress);
    CloseHandle(FileHandle);
```

```
    return CHECKSUM_SUCCESS;

} //MapFileAndCheckSumW()

//
// MapFileAndCheckSumA() - This functions maps the specified file and computes the checksum of the file.
//
// ARGUMENTS
//  Filename        Supplies a pointer to the name of the file whose checksum is computed.
//  HeaderSum       Supplies a pointer to a variable that receives the checksum from the image file,
//                  or zero if the file is not an image file.
//  CheckSum        Supplies a pointer to the variable that receive the computed checksum.
//
// RETURN VALUE     0 if successful, else error number.
//
// FROM "imagehlp/checksum.c": David N. Cutler 21-Mar-1993
//
ULONG MapFileAndCheckSumA(LPSTR Filename, LPDWORD HeaderSum, LPDWORD CheckSum)
{
    // Convert the file name to unicode and call the unicode version of this function.
    WCHAR FileNameW[MAX_PATH];
    if (MultiByteToWideChar(CP_ACP,MB_PRECOMPOSED,Filename,-1,FileNameW,MAX_PATH))
    {
        return MapFileAndCheckSumW(FileNameW,HeaderSum,CheckSum);
    }
    return CHECKSUM_UNICODE_FAILURE;

} //MapFileAndCheckSumA()
```

1:1