**MD5.h**

#pragma once

#define \_CRT\_SECURE\_NO\_WARNINGS

#include<fstream>

namespace MD5

{

typedef unsigned int uint;

typedef unsigned char uint8;

typedef unsigned long uint32;

typedef unsigned long long uint64;

class CMD5

{

public:

CMD5(

uint32 ui32MD5ChainingA = 0x67452301ul, //标准幻数A：(01234567)hex

uint32 ui32MD5ChainingB = 0xEFCDAB89ul, //标准幻数B：(89ABCDEF)hex

uint32 ui32MD5ChainingC = 0x98BADCFEul, //标准幻数C：(FEDCBA98)hex

uint32 ui32MD5ChainingD = 0x10325476ul //标准幻数D：(76543210)hex

);

//重置数据

inline void Reset();

//获取MD5值

inline char \* GetMD5(char \* szMD5) const;

//MD5：字节流

void CalcMD5(const void \* pvDataStream, uint32 ui32DataBytes);

//MD5：字符串

void CalcMD5(const char \* szString);

//MD5：文件

void CalcMD5(const char \* szFilePath, bool bIsBinary);

private:

static const uint msk\_uiMD5TransformShiftMatrix[4][4]; //MD5变换移位矩阵

static const uint32 msk\_ui32MD5TransformAddMatrix[4][16]; //MD5变换增量矩阵

static const uint8 msk\_ui8MD5PaddingArray[64]; //MD5拓展长度数组

uint32 m\_ui32MD5ChainingA; //MD5链接变量A

uint32 m\_ui32MD5ChainingB; //MD5链接变量B

uint32 m\_ui32MD5ChainingC; //MD5链接变量C

uint32 m\_ui32MD5ChainingD; //MD5链接变量D

uint32 m\_ui32State[4]; //MD5算法当前计算状态值

uint32 m\_ui32ByteCounts[2]; //数据总字节数

uint8 m\_ui8InputBuffer[64]; //输入缓冲区

char m\_szMD5[33]; //MD5计算结果

bool m\_bResetFlag; //复位标志

inline void Uint32to8(uint8 \* pui8Output, const uint32 \* pui32Input, uint uiInputByteCounts) const;

inline void Uint8to32(uint32 \* pui32Output, const uint8 \* pui8Input, uint uiInputByteCounts) const;

inline uint32 F(const uint32 x, const uint32 y, const uint32 z) const;

inline uint32 G(const uint32 x, const uint32 y, const uint32 z) const;

inline uint32 H(const uint32 x, const uint32 y, const uint32 z) const;

inline uint32 I(const uint32 x, const uint32 y, const uint32 z) const;

inline uint32 RotateLeft(const uint32 x, const uint uiShiftCount) const;

inline void FF(uint32 & a, uint32 b, uint32 c, uint32 d, uint32 x, uint s, uint32 ac);

inline void GG(uint32 & a, uint32 b, uint32 c, uint32 d, uint32 x, uint s, uint32 ac);

inline void HH(uint32 & a, uint32 b, uint32 c, uint32 d, uint32 x, uint s, uint32 ac);

inline void II(uint32 & a, uint32 b, uint32 c, uint32 d, uint32 x, uint s, uint32 ac);

void Transform(const uint8 ui8Block[64]);

void Update(const uint8 \* pui8Input, uint32 ui32InputBytes);

void Calculate();

};

//初始化MD5变换移位矩阵

const uint CMD5::msk\_uiMD5TransformShiftMatrix[4][4] =

{

{7, 12, 17, 22},

{5, 9, 14, 20},

{4, 11, 16, 23},

{6, 10, 15, 21}

};

//初始化MD5变换增量矩阵

const uint32 CMD5::msk\_ui32MD5TransformAddMatrix[4][16] =

{

/\*

loop i=1..64 : 4294967296 \* abs(sin(i))

4294967296 = 2^32

\*/

{

0xD76AA478, //0xd76aa478

0xE8C7B756, //0xe8c7b756

0x242070DB, //0x242070db

0xC1BDCEEE, //0xc1bdceee

0xF57C0FAF, //0xf57c0faf

0x4787C62A, //0x4787c62a

0xA8304613, //0xa8304613

0xFD469501, //0xfd469501

0x698098D8, //0x698098d8

0x8B44F7AF, //0x8b44f7af

0xFFFF5BB1, //0xffff5bb1

0x895CD7BE, //0x895cd7be

0x6B901122, //0x6b901122

0xFD987193, //0xfd987193

0xA679438E, //0xa679438e

0x49B40821 //0x49b40821

},

{

0xF61E2562, //0xf61e2562

0xC040B340, //0xc040b340

0x265E5A51, //0x265e5a51

0xE9B6C7AA, //0xe9b6c7aa

0xD62F105D, //0xd62f105d

0x02441453, //0x02441453

0xD8A1E681, //0xd8a1e681

0xE7D3FBC8, //0xe7d3fbc8

0x21E1CDE6, //0x21e1cde6

0xC33707D6, //0xc33707d6

0xF4D50D87, //0xf4d50d87

0x455A14ED, //0x455a14ed

0xA9E3E905, //0xa9e3e905

0xFCEFA3F8, //0xfcefa3f8

0x676F02D9, //0x676f02d9

0x8D2A4C8A //0x8d2a4c8a

},

{

0xFFFA3942, //0xfffa3942

0x8771F681, //0x8771f681

0x6D9D6122, //0x6d9d6122

0xFDE5380C, //0xfde5380c

0xA4BEEA44, //0xa4beea44

0x4BDECFA9, //0x4bdecfa9

0xF6BB4B60, //0xf6bb4b60

0xBEBFBC70, //0xbebfbc70

0x289B7EC6, //0x289b7ec6

0xEAA127FA, //0xeaa127fa

0xD4EF3085, //0xd4ef3085

0x04881D05, //0x04881d05

0xD9D4D039, //0xd9d4d039

0xE6DB99E5, //0xe6db99e5

0x1FA27CF8, //0x1fa27cf8

0xC4AC5665 //0xc4ac5665

},

{

0xF4292244, //0xf4292244

0x432AFF97, //0x432aff97

0xAB9423A7, //0xab9423a7

0xFC93A039, //0xfc93a039

0x655B59C3, //0x655b59c3

0x8F0CCC92, //0x8f0ccc92

0xFFEFF47D, //0xffeff47d

0x85845DD1, //0x85845dd1

0x6FA87E4F, //0x6fa87e4f

0xFE2CE6E0, //0xfe2ce6e0

0xA3014314, //0xa3014314

0x4E0811A1, //0x4e0811a1

0xF7537E82, //0xf7537e82

0xBD3AF235, //0xbd3af235

0x2AD7D2BB, //0x2ad7d2bb

0xEB86D391 //0xeb86d391

}

};

//初始化MD5拓展长度数组

const uint8 CMD5::msk\_ui8MD5PaddingArray[64] = { 0x80 };

//CMD5实现：

CMD5::CMD5(uint32 ui32MD5ChainingA, uint32 ui32MD5ChainingB, uint32 ui32MD5ChainingC, uint32 ui32MD5ChainingD)

{

m\_ui32MD5ChainingA = ui32MD5ChainingA;

m\_ui32MD5ChainingB = ui32MD5ChainingB;

m\_ui32MD5ChainingC = ui32MD5ChainingC;

m\_ui32MD5ChainingD = ui32MD5ChainingD;

this->Reset();

}

inline void CMD5::Reset()

{

m\_bResetFlag = true;

m\_ui32State[0] = m\_ui32MD5ChainingA;

m\_ui32State[1] = m\_ui32MD5ChainingB;

m\_ui32State[2] = m\_ui32MD5ChainingC;

m\_ui32State[3] = m\_ui32MD5ChainingD;

memset(m\_ui32ByteCounts, 0, sizeof(uint32) \* 2);

memset(m\_ui8InputBuffer, 0, sizeof(uint8) \* 64);

memset(m\_szMD5, 0, sizeof(char) \* 33);

return;

}

inline char \* CMD5::GetMD5(char \* szMD5) const

{

strcpy(szMD5, m\_szMD5);

return szMD5;

}

void CMD5::CalcMD5(const void \* pvDataStream, uint32 ui32DataBytes)

{

if (false == m\_bResetFlag)

{

return;

}

else

{

m\_bResetFlag = false;

}

this->Update((uint8 \*)(pvDataStream), ui32DataBytes);

this->Calculate();

return;

}

void CMD5::CalcMD5(const char \* szString)

{

if (false == m\_bResetFlag)

{

return;

}

else

{

m\_bResetFlag = false;

}

this->Update((uint8 \*)(szString), strlen(szString));

this->Calculate();

return;

}

void CMD5::CalcMD5(const char \* szFilePath, bool bIsBinary)

{

if (false == m\_bResetFlag)

{

return;

}

else

{

m\_bResetFlag = false;

}

std::ifstream fin(szFilePath, bIsBinary ? (std::ios::binary) : (std::ios\_base::in));

if (!fin)

{

return;

}

std::streamsize ReadBytes = 0;

char cReadBuffer[1024] = { 0 };

while (!fin.eof())

{

fin.read(cReadBuffer, 1024);

ReadBytes = fin.gcount();

if (ReadBytes > 0)

{

this->Update((uint8 \*)cReadBuffer, ReadBytes);

//streamsize-->uint32可能丢失数据，限制缓冲区大小为1024bytes

}

}

fin.close();

this->Calculate();

return;

}

inline void CMD5::Uint32to8(uint8 \* pui8Output, const uint32 \* pui32Input, uint uiInputByteCounts) const

{

for (uint uiInputPos = 0, uiOutputPos = 0; uiOutputPos < uiInputByteCounts; uiInputPos++)

{

pui8Output[uiOutputPos++] = (uint8)((pui32Input[uiInputPos]) & 0xFF);

pui8Output[uiOutputPos++] = (uint8)((pui32Input[uiInputPos] >> 8) & 0xFF);

pui8Output[uiOutputPos++] = (uint8)((pui32Input[uiInputPos] >> 16) & 0xFF);

pui8Output[uiOutputPos++] = (uint8)((pui32Input[uiInputPos] >> 24) & 0xFF);

}

return;

}

inline void CMD5::Uint8to32(uint32 \* pui32Output, const uint8 \* pui8Input, uint uiInputByteCounts) const

{

for (uint uiInputPos = 0, uiOutputPos = 0; uiInputPos < uiInputByteCounts; uiOutputPos++)

{

pui32Output[uiOutputPos] = (uint32)(pui8Input[uiInputPos++]);

pui32Output[uiOutputPos] |= (uint32)(pui8Input[uiInputPos++]) << 8;

pui32Output[uiOutputPos] |= (uint32)(pui8Input[uiInputPos++]) << 16;

pui32Output[uiOutputPos] |= (uint32)(pui8Input[uiInputPos++]) << 24;

}

return;

}

inline uint32 CMD5::F(const uint32 x, const uint32 y, const uint32 z) const

{

return ((x & y) | ((~x) & z));

}

inline uint32 CMD5::G(const uint32 x, const uint32 y, const uint32 z) const

{

return ((x & z) | (y & (~z)));

}

inline uint32 CMD5::H(const uint32 x, const uint32 y, const uint32 z) const

{

return x^y^z;

}

inline uint32 CMD5::I(const uint32 x, const uint32 y, const uint32 z) const

{

return (y ^ (x | (~z)));

}

inline uint32 CMD5::RotateLeft(const uint32 x, const uint uiShiftCount) const

{

return ((x << uiShiftCount) | (x >> (32 - uiShiftCount)));

}

inline void CMD5::FF(uint32 & a, uint32 b, uint32 c, uint32 d, uint32 x, uint s, uint32 ac)

{

a += F(b, c, d) + x + ac;

a = RotateLeft(a, s);

a += b;

}

inline void CMD5::GG(uint32 & a, uint32 b, uint32 c, uint32 d, uint32 x, uint s, uint32 ac)

{

a += G(b, c, d) + x + ac;

a = RotateLeft(a, s);

a += b;

}

inline void CMD5::HH(uint32 & a, uint32 b, uint32 c, uint32 d, uint32 x, uint s, uint32 ac)

{

a += H(b, c, d) + x + ac;

a = RotateLeft(a, s);

a += b;

}

inline void CMD5::II(uint32 & a, uint32 b, uint32 c, uint32 d, uint32 x, uint s, uint32 ac)

{

a += I(b, c, d) + x + ac;

a = RotateLeft(a, s);

a += b;

}

void CMD5::Transform(const uint8 ui8Block[64])

{

register uint32 a = m\_ui32State[0];

register uint32 b = m\_ui32State[1];

register uint32 c = m\_ui32State[2];

register uint32 d = m\_ui32State[3];

uint32 ui32Block[16];

this->Uint8to32(ui32Block, ui8Block, 64);

/\* Round 1 \*/

FF(a, b, c, d, ui32Block[ 0], msk\_uiMD5TransformShiftMatrix[0][0], msk\_ui32MD5TransformAddMatrix[0][ 0]); /\* 1 \*/

FF(d, a, b, c, ui32Block[ 1], msk\_uiMD5TransformShiftMatrix[0][1], msk\_ui32MD5TransformAddMatrix[0][ 1]); /\* 2 \*/

FF(c, d, a, b, ui32Block[ 2], msk\_uiMD5TransformShiftMatrix[0][2], msk\_ui32MD5TransformAddMatrix[0][ 2]); /\* 3 \*/

FF(b, c, d, a, ui32Block[ 3], msk\_uiMD5TransformShiftMatrix[0][3], msk\_ui32MD5TransformAddMatrix[0][ 3]); /\* 4 \*/

FF(a, b, c, d, ui32Block[ 4], msk\_uiMD5TransformShiftMatrix[0][0], msk\_ui32MD5TransformAddMatrix[0][ 4]); /\* 5 \*/

FF(d, a, b, c, ui32Block[ 5], msk\_uiMD5TransformShiftMatrix[0][1], msk\_ui32MD5TransformAddMatrix[0][ 5]); /\* 6 \*/

FF(c, d, a, b, ui32Block[ 6], msk\_uiMD5TransformShiftMatrix[0][2], msk\_ui32MD5TransformAddMatrix[0][ 6]); /\* 7 \*/

FF(b, c, d, a, ui32Block[ 7], msk\_uiMD5TransformShiftMatrix[0][3], msk\_ui32MD5TransformAddMatrix[0][ 7]); /\* 8 \*/

FF(a, b, c, d, ui32Block[ 8], msk\_uiMD5TransformShiftMatrix[0][0], msk\_ui32MD5TransformAddMatrix[0][ 8]); /\* 9 \*/

FF(d, a, b, c, ui32Block[ 9], msk\_uiMD5TransformShiftMatrix[0][1], msk\_ui32MD5TransformAddMatrix[0][ 9]); /\* 10 \*/

FF(c, d, a, b, ui32Block[10], msk\_uiMD5TransformShiftMatrix[0][2], msk\_ui32MD5TransformAddMatrix[0][10]); /\* 11 \*/

FF(b, c, d, a, ui32Block[11], msk\_uiMD5TransformShiftMatrix[0][3], msk\_ui32MD5TransformAddMatrix[0][11]); /\* 12 \*/

FF(a, b, c, d, ui32Block[12], msk\_uiMD5TransformShiftMatrix[0][0], msk\_ui32MD5TransformAddMatrix[0][12]); /\* 13 \*/

FF(d, a, b, c, ui32Block[13], msk\_uiMD5TransformShiftMatrix[0][1], msk\_ui32MD5TransformAddMatrix[0][13]); /\* 14 \*/

FF(c, d, a, b, ui32Block[14], msk\_uiMD5TransformShiftMatrix[0][2], msk\_ui32MD5TransformAddMatrix[0][14]); /\* 15 \*/

FF(b, c, d, a, ui32Block[15], msk\_uiMD5TransformShiftMatrix[0][3], msk\_ui32MD5TransformAddMatrix[0][15]); /\* 16 \*/

/\* Round 2 \*/

GG(a, b, c, d, ui32Block[ 1], msk\_uiMD5TransformShiftMatrix[1][0], msk\_ui32MD5TransformAddMatrix[1][ 0]); /\* 17 \*/

GG(d, a, b, c, ui32Block[ 6], msk\_uiMD5TransformShiftMatrix[1][1], msk\_ui32MD5TransformAddMatrix[1][ 1]); /\* 18 \*/

GG(c, d, a, b, ui32Block[11], msk\_uiMD5TransformShiftMatrix[1][2], msk\_ui32MD5TransformAddMatrix[1][ 2]); /\* 19 \*/

GG(b, c, d, a, ui32Block[ 0], msk\_uiMD5TransformShiftMatrix[1][3], msk\_ui32MD5TransformAddMatrix[1][ 3]); /\* 20 \*/

GG(a, b, c, d, ui32Block[ 5], msk\_uiMD5TransformShiftMatrix[1][0], msk\_ui32MD5TransformAddMatrix[1][ 4]); /\* 21 \*/

GG(d, a, b, c, ui32Block[10], msk\_uiMD5TransformShiftMatrix[1][1], msk\_ui32MD5TransformAddMatrix[1][ 5]); /\* 22 \*/

GG(c, d, a, b, ui32Block[15], msk\_uiMD5TransformShiftMatrix[1][2], msk\_ui32MD5TransformAddMatrix[1][ 6]); /\* 23 \*/

GG(b, c, d, a, ui32Block[ 4], msk\_uiMD5TransformShiftMatrix[1][3], msk\_ui32MD5TransformAddMatrix[1][ 7]); /\* 24 \*/

GG(a, b, c, d, ui32Block[ 9], msk\_uiMD5TransformShiftMatrix[1][0], msk\_ui32MD5TransformAddMatrix[1][ 8]); /\* 25 \*/

GG(d, a, b, c, ui32Block[14], msk\_uiMD5TransformShiftMatrix[1][1], msk\_ui32MD5TransformAddMatrix[1][ 9]); /\* 26 \*/

GG(c, d, a, b, ui32Block[ 3], msk\_uiMD5TransformShiftMatrix[1][2], msk\_ui32MD5TransformAddMatrix[1][10]); /\* 27 \*/

GG(b, c, d, a, ui32Block[ 8], msk\_uiMD5TransformShiftMatrix[1][3], msk\_ui32MD5TransformAddMatrix[1][11]); /\* 28 \*/

GG(a, b, c, d, ui32Block[13], msk\_uiMD5TransformShiftMatrix[1][0], msk\_ui32MD5TransformAddMatrix[1][12]); /\* 29 \*/

GG(d, a, b, c, ui32Block[ 2], msk\_uiMD5TransformShiftMatrix[1][1], msk\_ui32MD5TransformAddMatrix[1][13]); /\* 30 \*/

GG(c, d, a, b, ui32Block[ 7], msk\_uiMD5TransformShiftMatrix[1][2], msk\_ui32MD5TransformAddMatrix[1][14]); /\* 31 \*/

GG(b, c, d, a, ui32Block[12], msk\_uiMD5TransformShiftMatrix[1][3], msk\_ui32MD5TransformAddMatrix[1][15]); /\* 32 \*/

/\* Round 3 \*/

HH(a, b, c, d, ui32Block[ 5], msk\_uiMD5TransformShiftMatrix[2][0], msk\_ui32MD5TransformAddMatrix[2][ 0]); /\* 33 \*/

HH(d, a, b, c, ui32Block[ 8], msk\_uiMD5TransformShiftMatrix[2][1], msk\_ui32MD5TransformAddMatrix[2][ 1]); /\* 34 \*/

HH(c, d, a, b, ui32Block[11], msk\_uiMD5TransformShiftMatrix[2][2], msk\_ui32MD5TransformAddMatrix[2][ 2]); /\* 35 \*/

HH(b, c, d, a, ui32Block[14], msk\_uiMD5TransformShiftMatrix[2][3], msk\_ui32MD5TransformAddMatrix[2][ 3]); /\* 36 \*/

HH(a, b, c, d, ui32Block[ 1], msk\_uiMD5TransformShiftMatrix[2][0], msk\_ui32MD5TransformAddMatrix[2][ 4]); /\* 37 \*/

HH(d, a, b, c, ui32Block[ 4], msk\_uiMD5TransformShiftMatrix[2][1], msk\_ui32MD5TransformAddMatrix[2][ 5]); /\* 38 \*/

HH(c, d, a, b, ui32Block[ 7], msk\_uiMD5TransformShiftMatrix[2][2], msk\_ui32MD5TransformAddMatrix[2][ 6]); /\* 39 \*/

HH(b, c, d, a, ui32Block[10], msk\_uiMD5TransformShiftMatrix[2][3], msk\_ui32MD5TransformAddMatrix[2][ 7]); /\* 40 \*/

HH(a, b, c, d, ui32Block[13], msk\_uiMD5TransformShiftMatrix[2][0], msk\_ui32MD5TransformAddMatrix[2][ 8]); /\* 41 \*/

HH(d, a, b, c, ui32Block[ 0], msk\_uiMD5TransformShiftMatrix[2][1], msk\_ui32MD5TransformAddMatrix[2][ 9]); /\* 42 \*/

HH(c, d, a, b, ui32Block[ 3], msk\_uiMD5TransformShiftMatrix[2][2], msk\_ui32MD5TransformAddMatrix[2][10]); /\* 43 \*/

HH(b, c, d, a, ui32Block[ 6], msk\_uiMD5TransformShiftMatrix[2][3], msk\_ui32MD5TransformAddMatrix[2][11]); /\* 44 \*/

HH(a, b, c, d, ui32Block[ 9], msk\_uiMD5TransformShiftMatrix[2][0], msk\_ui32MD5TransformAddMatrix[2][12]); /\* 45 \*/

HH(d, a, b, c, ui32Block[12], msk\_uiMD5TransformShiftMatrix[2][1], msk\_ui32MD5TransformAddMatrix[2][13]); /\* 46 \*/

HH(c, d, a, b, ui32Block[15], msk\_uiMD5TransformShiftMatrix[2][2], msk\_ui32MD5TransformAddMatrix[2][14]); /\* 47 \*/

HH(b, c, d, a, ui32Block[ 2], msk\_uiMD5TransformShiftMatrix[2][3], msk\_ui32MD5TransformAddMatrix[2][15]); /\* 48 \*/

/\* Round 4 \*/

II(a, b, c, d, ui32Block[ 0], msk\_uiMD5TransformShiftMatrix[3][0], msk\_ui32MD5TransformAddMatrix[3][ 0]); /\* 49 \*/

II(d, a, b, c, ui32Block[ 7], msk\_uiMD5TransformShiftMatrix[3][1], msk\_ui32MD5TransformAddMatrix[3][ 1]); /\* 50 \*/

II(c, d, a, b, ui32Block[14], msk\_uiMD5TransformShiftMatrix[3][2], msk\_ui32MD5TransformAddMatrix[3][ 2]); /\* 51 \*/

II(b, c, d, a, ui32Block[ 5], msk\_uiMD5TransformShiftMatrix[3][3], msk\_ui32MD5TransformAddMatrix[3][ 3]); /\* 52 \*/

II(a, b, c, d, ui32Block[12], msk\_uiMD5TransformShiftMatrix[3][0], msk\_ui32MD5TransformAddMatrix[3][ 4]); /\* 53 \*/

II(d, a, b, c, ui32Block[ 3], msk\_uiMD5TransformShiftMatrix[3][1], msk\_ui32MD5TransformAddMatrix[3][ 5]); /\* 54 \*/

II(c, d, a, b, ui32Block[10], msk\_uiMD5TransformShiftMatrix[3][2], msk\_ui32MD5TransformAddMatrix[3][ 6]); /\* 55 \*/

II(b, c, d, a, ui32Block[ 1], msk\_uiMD5TransformShiftMatrix[3][3], msk\_ui32MD5TransformAddMatrix[3][ 7]); /\* 56 \*/

II(a, b, c, d, ui32Block[ 8], msk\_uiMD5TransformShiftMatrix[3][0], msk\_ui32MD5TransformAddMatrix[3][ 8]); /\* 57 \*/

II(d, a, b, c, ui32Block[15], msk\_uiMD5TransformShiftMatrix[3][1], msk\_ui32MD5TransformAddMatrix[3][ 9]); /\* 58 \*/

II(c, d, a, b, ui32Block[ 6], msk\_uiMD5TransformShiftMatrix[3][2], msk\_ui32MD5TransformAddMatrix[3][10]); /\* 59 \*/

II(b, c, d, a, ui32Block[13], msk\_uiMD5TransformShiftMatrix[3][3], msk\_ui32MD5TransformAddMatrix[3][11]); /\* 60 \*/

II(a, b, c, d, ui32Block[ 4], msk\_uiMD5TransformShiftMatrix[3][0], msk\_ui32MD5TransformAddMatrix[3][12]); /\* 61 \*/

II(d, a, b, c, ui32Block[11], msk\_uiMD5TransformShiftMatrix[3][1], msk\_ui32MD5TransformAddMatrix[3][13]); /\* 62 \*/

II(c, d, a, b, ui32Block[ 2], msk\_uiMD5TransformShiftMatrix[3][2], msk\_ui32MD5TransformAddMatrix[3][14]); /\* 63 \*/

II(b, c, d, a, ui32Block[ 9], msk\_uiMD5TransformShiftMatrix[3][3], msk\_ui32MD5TransformAddMatrix[3][15]); /\* 64 \*/

m\_ui32State[0] += a;

m\_ui32State[1] += b;

m\_ui32State[2] += c;

m\_ui32State[3] += d;

return;

}

void CMD5::Update(const uint8 \* pui8Input, uint32 ui32InputBytes)

{

//计算m\_ui32ByteCounts[0]对64的余数，即：uiPos = m\_ui32ByteCounts[0] mod 64

uint uiMod = (uint)((m\_ui32ByteCounts[0] >> 3) & 0x3F); /\* (0x3F)hex = (63)dec = (0011,1111)bin \*/

//更新字节数

if ((m\_ui32ByteCounts[0] += (ui32InputBytes << 3)) < (ui32InputBytes << 3))

{

m\_ui32ByteCounts[1]++;

}

m\_ui32ByteCounts[1] += (ui32InputBytes >> 29);

//尽可能多的进行MD5变换

uint32 ui32 = 0;

uint uiPartBytes = 64 - uiMod;

if (ui32InputBytes >= uiPartBytes)

{

memcpy(&m\_ui8InputBuffer[uiMod], pui8Input, uiPartBytes);

this->Transform(m\_ui8InputBuffer);

for (ui32 = uiPartBytes; (ui32 + 63) < ui32InputBytes; ui32 += 64)

{

this->Transform(&pui8Input[ui32]);

}

}

//输入区内剩余的数据

memcpy(&m\_ui8InputBuffer[uiMod], &pui8Input[ui32], ui32InputBytes - ui32);

return;

}

void CMD5::Calculate()

{

//整个数据流的字节数

uint8 ui8TotalBytes[8] = { 0 };

//转换uint类型

this->Uint32to8(ui8TotalBytes, m\_ui32ByteCounts, 8); //2 \* sizeof(uint32)

//拓展字节到 56 mod 64

uint uiMod = (uint)((m\_ui32ByteCounts[0] >> 3) & 0x3f);

uint uiPad = (uiMod < 56) ? (56 - uiMod) : (120 - uiMod);

this->Update(msk\_ui8MD5PaddingArray, uiPad);

//附加数据流的字节大小的数据

this->Update(ui8TotalBytes, 8);

//处理MD5字符串

uint8 ui8MD5Str[16] = { 0 };

this->Uint32to8(ui8MD5Str, m\_ui32State, 16); //4 \* sizeof(uint32)

for (uint ui = 0; ui < 16; ui++)

{

if (ui8MD5Str[ui] == 0)

{

strcat(m\_szMD5, "00");

}

else if (ui8MD5Str[ui] <= 15)

{

sprintf(m\_szMD5, "%s0%X", m\_szMD5, ui8MD5Str[ui]);

}

else

{

sprintf(m\_szMD5, "%s%X", m\_szMD5, ui8MD5Str[ui]);

}

}

return;

}

}

**NotifyIcon.h**

#pragma once

#include<Windows.h>

#include<CommCtrl.h>

namespace NOTIFYICON

{

class CNotifyIcon

{

public:

CNotifyIcon(HWND hOwnerWnd, UINT uiID, UINT uiCallbackMessage, bool bHighVersion = false);

inline void SetValidParamFlags(UINT uiFlags);

inline void SetOwnerWindow(HWND hwnd);

inline void SetID(UINT uiID);

inline void SetCallbackMessageCode(UINT uiCallbackMessage);

inline void SetIcon(HINSTANCE hInst, PCTSTR szIconResourceName);

inline void SetTip(PCTSTR szTip);

inline void SetBalloonTimeout(UINT uiTimeoutMS);

inline void SetBalloonTitle(PCTSTR szTitle = nullptr);

inline void SetBalloonInfo(PCTSTR szInfo = nullptr);

inline void SetBalloonIcon(DWORD dwFlags, HINSTANCE hInst = NULL, PCTSTR szIconResourceName = nullptr);

inline void SetBalloonLargeIcon(HINSTANCE hInst = NULL, PCWSTR szIconResourceName = MAKEINTRESOURCEW(IDI\_APPLICATION));

inline BOOL AddNotificationIcon();

inline BOOL DeleteNotificationIcon();

inline BOOL ShowBalloon();

private:

NOTIFYICONDATA m\_nid;

};

//CNotifyIcon实现：

CNotifyIcon::CNotifyIcon(HWND hOwnerWnd, UINT uiID, UINT uiCallbackMessage, bool bHighVersion)

{

memset(&m\_nid, 0, sizeof(NOTIFYICONDATA));

m\_nid.cbSize = sizeof(NOTIFYICONDATA);

m\_nid.hWnd = hOwnerWnd;

m\_nid.uID = uiID;

m\_nid.uCallbackMessage = uiCallbackMessage;

m\_nid.uFlags = NIF\_ICON | NIF\_MESSAGE | NIF\_TIP;

m\_nid.hIcon = LoadIcon(NULL, MAKEINTRESOURCE(IDI\_APPLICATION));

wsprintf(m\_nid.szTip, TEXT("标准工具提示信息。"));

if (bHighVersion) //Version 5.0 (Version 6.0: NIIF\_NOSOUND)

{

m\_nid.uFlags |= NIF\_INFO;

//气球提示超时时间（10000ms-30000ms）

m\_nid.uTimeout = 15000;

//气球提示标题

wsprintf(m\_nid.szInfoTitle, TEXT("气球提示标题"));

//气球提示内容（是否显示气球和立即去除气球通过m\_nid.szInfo的值和NIM\_MODIFY设置）

wsprintf(m\_nid.szInfo, TEXT("气球提示内容。"));

//气球提示框图标（若m\_nid.szInfoTitle为空字符串，图标不显示）

m\_nid.dwInfoFlags = NIIF\_INFO;

}

}

inline void CNotifyIcon::SetValidParamFlags(UINT uiFlags)

{

m\_nid.uFlags = uiFlags;

}

inline void CNotifyIcon::SetOwnerWindow(HWND hwnd)

{

m\_nid.hWnd = hwnd;

}

inline void CNotifyIcon::SetID(UINT uiID)

{

m\_nid.uID = uiID;

}

inline void CNotifyIcon::SetCallbackMessageCode(UINT uiCallbackMessage)

{

m\_nid.uCallbackMessage = uiCallbackMessage;

}

inline void CNotifyIcon::SetIcon(HINSTANCE hInst, PCTSTR szIconResourceName)

{

m\_nid.hIcon = LoadIcon(hInst, szIconResourceName);

}

inline void CNotifyIcon::SetTip(PCTSTR szTip)

{

wsprintf(m\_nid.szTip, szTip);

}

inline void CNotifyIcon::SetBalloonTimeout(UINT uiTimeoutMS)

{

m\_nid.uTimeout = uiTimeoutMS;

}

inline void CNotifyIcon::SetBalloonTitle(PCTSTR szTitle)

{

if (szTitle)

{

wsprintf(m\_nid.szInfoTitle, szTitle);

}

else

{

wsprintf(m\_nid.szInfoTitle, TEXT(""));

}

}

inline void CNotifyIcon::SetBalloonInfo(PCTSTR szInfo)

{

if (szInfo)

{

wsprintf(m\_nid.szInfo, szInfo);

}

else

{

wsprintf(m\_nid.szInfo, TEXT(""));

}

}

inline void CNotifyIcon::SetBalloonIcon(DWORD dwFlags, HINSTANCE hInst, PCTSTR szIconResourceName)

{

m\_nid.dwInfoFlags = dwFlags;

if (dwFlags & NIIF\_USER)

{

m\_nid.hBalloonIcon = LoadIcon(hInst, szIconResourceName);

}

}

inline void CNotifyIcon::SetBalloonLargeIcon(HINSTANCE hInst, PCWSTR szIconResourceName)

{

m\_nid.dwInfoFlags = NIIF\_USER | NIIF\_LARGE\_ICON;

LoadIconMetric(hInst, szIconResourceName, LIM\_LARGE, &m\_nid.hBalloonIcon);

}

inline BOOL CNotifyIcon::AddNotificationIcon()

{

return Shell\_NotifyIcon(NIM\_ADD, &m\_nid);

}

inline BOOL CNotifyIcon::DeleteNotificationIcon()

{

return Shell\_NotifyIcon(NIM\_DELETE, &m\_nid);

}

inline BOOL CNotifyIcon::ShowBalloon()

{

//要求启用高版本

//更改m\_nid的：dwInfoFlags（气泡图标选择子）、szInfoTitle（气泡标题）、szInfo（气泡内容）

return Shell\_NotifyIcon(NIM\_MODIFY, &m\_nid);

}

}

**CmnDlg.h**

#pragma once

#include<Windows.h>

namespace COMMONDIALOG

{

class COSDlg

{

public:

COSDlg(HWND hOwnerWnd);

~COSDlg();

inline void SetOwnerWindow(HWND hwnd);

void SetTitle(LPCTSTR szTitle = nullptr);

void SetFilter(UINT uiExtCount, ...);

void SetDefExt(const TCHAR szDefExt[4] = nullptr);

inline const TCHAR \* GetFilePath() const;

inline DWORD GetFilterIndex() const;

inline WORD GetFileNameOffset() const;

inline WORD GetFileExtOffset() const;

inline BOOL CmnDlgOpenFile();

inline BOOL CmnDlgSaveFile();

private:

OPENFILENAME m\_ofn;

TCHAR \* m\_szFilter;

TCHAR \* m\_szTitle;

TCHAR \* m\_szDefExt;

TCHAR m\_szFilePath [MAX\_PATH];

};

class CBrowseFolderDlg

{

public:

CBrowseFolderDlg::CBrowseFolderDlg(HWND hOwnerWnd);

inline void SetOwnerWindow(HWND hwnd);

inline bool CmmDlgBrowse();

inline const TCHAR \* GetDir() const;

private:

BROWSEINFO m\_bi;

TCHAR m\_szSelectDir[MAX\_PATH];

TCHAR m\_szTitle[64];

};

//COSDlg实现：

COSDlg::COSDlg(HWND hOwnerWnd)

{

m\_szFilter = nullptr;

m\_szTitle = nullptr;

m\_szDefExt = nullptr;

memset(m\_szFilePath, 0, sizeof(m\_szFilePath));

//--------------------- 初始化ofn ---------------------//

memset(&m\_ofn, 0, sizeof(OPENFILENAME));

m\_ofn.lStructSize = sizeof(OPENFILENAME);

//---------- 使用的参数 ----------//

m\_ofn.hwndOwner = hOwnerWnd; //[in]------打开、保存对话框父窗口

m\_ofn.Flags = 0; //[in]------功能选择子（在函数中具体设置）

m\_ofn.lpstrTitle = nullptr; //[in]------打开、保存对话框窗口标题栏字符串（独立函数设置）

m\_ofn.lpstrFilter = nullptr; //[in]------筛选器字符串（独立函数设置）

m\_ofn.nFilterIndex = 0; //[in/out]--控件中当前选择的筛选器的索引

m\_ofn.lpstrFile = m\_szFilePath; //[in/out]--控件当前获取的文件及其路径

m\_ofn.nMaxFile = MAX\_PATH; //[in]------ofn.lpstrFile的最大字符个数

m\_ofn.lpstrDefExt = nullptr; //[in]------默认扩展名（独立函数设置）

m\_ofn.nFileOffset = 0; //[out]-----从路径开始到文件名的偏移字符个数

m\_ofn.nFileExtension = 0; //[out]-----从路径开始到文件名的扩展名的偏移字符个数

//--------- 未启用的参数 ---------//

m\_ofn.hInstance = NULL; //[in]---对话框模板对象实例句柄

m\_ofn.lpTemplateName = nullptr; //[in]---对话框模板资源的名字

m\_ofn.lpstrCustomFilter = nullptr; //[out]--定制筛选器缓冲区指针

m\_ofn.nMaxCustFilter = 0; //[in]---ofn.lpstrCustomFilter的最大字符个数

m\_ofn.lpstrTitle = nullptr; //[out]--控件当前获取的文件的文件名

m\_ofn.nMaxFileTitle = 0; //[in]---ofn.lpstrTitle的最大字符个数

m\_ofn.lpstrInitialDir = nullptr; //[in]---打开、保存对话框的初始目录

m\_ofn.lpfnHook = nullptr; //[in]---钩子函数

m\_ofn.lCustData = 0; //[in]---传递到钩子函数的数据

}

COSDlg::~COSDlg()

{

if (m\_szFilter)

{

delete [] m\_szFilter;

m\_szFilter = nullptr;

}

if (m\_szTitle)

{

delete [] m\_szTitle;

m\_szTitle = nullptr;

}

if (m\_szDefExt)

{

delete [] m\_szDefExt;

m\_szDefExt = nullptr;

}

}

inline void COSDlg::SetOwnerWindow(HWND hwnd)

{

m\_ofn.hwndOwner = hwnd;

}

void COSDlg::SetTitle(LPCTSTR szTitle)

{

if (m\_szTitle)

{

delete [] m\_szTitle;

}

if (szTitle)

{

m\_szTitle = new TCHAR [lstrlen(szTitle) + 1];

lstrcpy(m\_szTitle, szTitle);

}

else

{

m\_szTitle = nullptr;

}

m\_ofn.lpstrTitle = m\_szTitle;

}

void COSDlg::SetFilter(UINT uiExtCount, ...)

{

if (m\_szFilter)

{

delete [] m\_szFilter;

}

if (!uiExtCount)

{

m\_szFilter = nullptr;

m\_ofn.lpstrFilter = m\_szFilter;

return;

}

UINT uiTotalStringLen = 0;

TCHAR \*\* pszFilterPart = new TCHAR \* [2 \* uiExtCount];

va\_list arg\_ptr;

va\_start(arg\_ptr, uiExtCount);

for (UINT ui = 0; ui < 2 \* uiExtCount; ui++)

{

pszFilterPart[ui] = va\_arg(arg\_ptr, TCHAR \*);

uiTotalStringLen += lstrlen(pszFilterPart[ui]) + 1;

}

uiTotalStringLen++;

va\_end(arg\_ptr);

m\_szFilter = new TCHAR [uiTotalStringLen];

memset(m\_szFilter, '\0', uiTotalStringLen \* sizeof(TCHAR));

TCHAR \* pPos = &m\_szFilter[0];

for (UINT ui = 0; ui < 2 \* uiExtCount; ui++)

{

lstrcat(pPos, pszFilterPart[ui]);

pPos += lstrlen(pszFilterPart[ui]) + 1;

}

m\_ofn.lpstrFilter = m\_szFilter;

}

void COSDlg::SetDefExt(const TCHAR szDefExt[4])

{

if (szDefExt)

{

if (!m\_szDefExt)

{

m\_szDefExt = new TCHAR [4];

}

lstrcpy(m\_szDefExt, szDefExt);

}

else

{

if (m\_szDefExt)

{

delete [] m\_szDefExt;

m\_szDefExt = nullptr;

}

}

m\_ofn.lpstrDefExt = m\_szDefExt;

}

inline const TCHAR \* COSDlg::GetFilePath() const

{

return m\_szFilePath;

}

inline DWORD COSDlg::GetFilterIndex() const

{

return m\_ofn.nFilterIndex;

}

inline WORD COSDlg::GetFileNameOffset() const

{

return m\_ofn.nFileOffset;

}

inline WORD COSDlg::GetFileExtOffset() const

{

return m\_ofn.nFileExtension;

}

inline BOOL COSDlg::CmnDlgOpenFile()

{

m\_ofn.Flags = OFN\_EXPLORER | OFN\_HIDEREADONLY | OFN\_FILEMUSTEXIST;

return GetOpenFileName(&m\_ofn);

}

inline BOOL COSDlg::CmnDlgSaveFile()

{

m\_ofn.Flags = OFN\_EXPLORER | OFN\_CREATEPROMPT | OFN\_OVERWRITEPROMPT;

return GetSaveFileName(&m\_ofn);

}

//CBrowseFolderDlg实现：

CBrowseFolderDlg::CBrowseFolderDlg(HWND hOwnerWnd)

{

m\_bi.hwndOwner = hOwnerWnd;

m\_bi.pidlRoot = NULL;

m\_bi.pszDisplayName = m\_szSelectDir;

m\_bi.lpszTitle = m\_szTitle;

m\_bi.ulFlags = /\*BIF\_RETURNONLYFSDIRS\*/0;

m\_bi.lpfn = NULL;

m\_bi.lParam = 0;

m\_bi.iImage = 0;

wsprintf(m\_szTitle, TEXT("选择文件夹："));

}

inline void CBrowseFolderDlg::SetOwnerWindow(HWND hwnd)

{

m\_bi.hwndOwner = hwnd;

}

inline bool CBrowseFolderDlg::CmmDlgBrowse()

{

LPITEMIDLIST pItemIDList = SHBrowseForFolder(&m\_bi);

if (!pItemIDList)

{

return false;

}

if (!SHGetPathFromIDList(pItemIDList, m\_szSelectDir))

{

//获取文件夹信息失败

//MessageBox(NULL, TEXT("获取文件夹信息失败！"), TEXT("错误"), MB\_ICONERROR);

}

//使用IMalloc接口防止内存泄漏

IMalloc \* pMalloc;

if (NOERROR != SHGetMalloc(&pMalloc))

{

//未返回有效的IMalloc接口指针,无法取得外壳程序的IMalloc接口

//MessageBox(NULL, TEXT("释放接口指针失败！"), TEXT("错误"), MB\_ICONERROR);

}

pMalloc->Free(pItemIDList);

if (pMalloc)

{

pMalloc->Release();

}

}

inline const TCHAR \* CBrowseFolderDlg::GetDir() const

{

return m\_szSelectDir;

}

}

**AutoDiskCopier\_pre.h**

#pragma once

#define \_CRT\_SECURE\_NO\_WARNINGS

#include<Windows.h>

#include<process.h>

#include<Dbt.h>

#include<Shlwapi.h>

#pragma comment(lib, "shlwapi.lib")

#include<shlobj.h>

#pragma comment(lib, "shell32.lib")

#include<imagehlp.h>

#pragma comment(lib, "imagehlp.lib")

#include"resource.h"

#include"CmnDlg.h"

#include"NotifyIcon.h"

#include"MD5.h"

using namespace COMMONDIALOG;

using namespace NOTIFYICON;

using namespace MD5;

#define WM\_NOTIFYICON (WM\_USER + 1)

class CAutoDiskCopier

{

public:

CMD5 m\_md5;

COSDlg m\_osdlg;

CBrowseFolderDlg m\_bfdlg;

CNotifyIcon m\_ni;

char m\_szSettingFilePath[MAX\_PATH];

bool m\_bEnableNotifyIcon;

bool m\_bMD5Check;

bool m\_bPopupConfirmDialog;

bool m\_bAutorun;

TCHAR m\_szCopyFileSavePath[MAX\_PATH];

bool m\_bCheckKeyFile;

TCHAR m\_szKey[MAX\_PATH];

CAutoDiskCopier(HWND hwnd) : m\_md5(), m\_osdlg(NULL), m\_bfdlg(NULL), m\_ni(hwnd, IDI\_ICON, WM\_NOTIFYICON, false)

{

SHGetSpecialFolderPathA(hwnd, m\_szSettingFilePath, 26/\*“C:/Program Files”\*/, TRUE);

sprintf(m\_szSettingFilePath, "%s%s", m\_szSettingFilePath, "\\AutoDiskCopier\\Setting\\");

MakeSureDirectoryPathExists(m\_szSettingFilePath);

sprintf(m\_szSettingFilePath, "%s%s", m\_szSettingFilePath, "ADC\_Setting.set");

HANDLE hFile = CreateFileA(m\_szSettingFilePath, GENERIC\_READ, 0, NULL, OPEN\_EXISTING, FILE\_ATTRIBUTE\_NORMAL, NULL);

if (INVALID\_HANDLE\_VALUE == hFile)

{

//打开配置文件失败 -> 初始化

hFile = CreateFileA(m\_szSettingFilePath, GENERIC\_WRITE, 0, NULL, CREATE\_ALWAYS, FILE\_ATTRIBUTE\_NORMAL, NULL);

if (INVALID\_HANDLE\_VALUE == hFile)

{

//初始化失败 -> 程序退出

MessageBox(NULL, TEXT("程序初始化失败！\n创建配置文件失败！"), TEXT("ADC - 初始化错误"), MB\_ICONERROR);

PostQuitMessage(GetLastError());

}

else

{

//初始化成功

SHGetSpecialFolderPath(hwnd, m\_szCopyFileSavePath, 26/\*“C:/Program Files”\*/, TRUE);

wsprintf(m\_szCopyFileSavePath, TEXT("%s%s"), m\_szCopyFileSavePath, TEXT("\\AutoDiskCopier\\CopiedFiles\\"));

CreateDirectory(m\_szCopyFileSavePath, NULL);

m\_bEnableNotifyIcon = true;

m\_bAutorun = false;

m\_bPopupConfirmDialog = true;

m\_bMD5Check = false;

m\_bCheckKeyFile = false;

wsprintf(m\_szKey, TEXT(""));

DWORD dwWrited = 0;

WriteFile(hFile, &m\_bEnableNotifyIcon, sizeof(bool), &dwWrited, NULL);

WriteFile(hFile, &m\_bAutorun, sizeof(bool), &dwWrited, NULL);

WriteFile(hFile, &m\_bPopupConfirmDialog, sizeof(bool), &dwWrited, NULL);

WriteFile(hFile, &m\_bMD5Check, sizeof(bool), &dwWrited, NULL);

WriteFile(hFile, &m\_bCheckKeyFile, sizeof(bool), &dwWrited, NULL);

WriteFile(hFile, m\_szCopyFileSavePath, MAX\_PATH \* sizeof(TCHAR), &dwWrited, NULL);

WriteFile(hFile, m\_szKey, MAX\_PATH \* sizeof(TCHAR), &dwWrited, NULL);

CloseHandle(hFile);

MessageBox(NULL, TEXT("程序初始化完成。"), TEXT("ADC - 初始化完成"), MB\_OK);

}

}

else

{

//打开配置文件成功

DWORD dwReaded = 0;

ReadFile(hFile, &m\_bEnableNotifyIcon, sizeof(bool), &dwReaded, NULL);

ReadFile(hFile, &m\_bAutorun, sizeof(bool), &dwReaded, NULL);

ReadFile(hFile, &m\_bPopupConfirmDialog, sizeof(bool), &dwReaded, NULL);

ReadFile(hFile, &m\_bMD5Check, sizeof(bool), &dwReaded, NULL);

ReadFile(hFile, &m\_bCheckKeyFile, sizeof(bool), &dwReaded, NULL);

ReadFile(hFile, m\_szCopyFileSavePath, MAX\_PATH \* sizeof(TCHAR), &dwReaded, NULL);

ReadFile(hFile, m\_szKey, MAX\_PATH \* sizeof(TCHAR), &dwReaded, NULL);

CloseHandle(hFile);

}

}

void Copying(PCTSTR szCopyDiskPath)

{

if (m\_bPopupConfirmDialog)

{

if (IDYES != MessageBox(NULL, TEXT("已检测到磁盘接入，是否执行复制？"), TEXT("ADC - 确认复制"), MB\_YESNO))

{

return;

}

}

/\*\_beginthread(&(CAutoDiskCopier::CopyingThread), 0, (void \*)szCopyDiskPath);\*/

CopyingThread(szCopyDiskPath); //

}

void SetDlgInfo(HWND hwnd)

{

CheckDlgButton(hwnd, IDC\_CHCK\_SHOWNOTIFYICON, m\_bEnableNotifyIcon ? BST\_CHECKED : BST\_UNCHECKED);

CheckDlgButton(hwnd, IDC\_CHCK\_MD5, m\_bMD5Check ? BST\_CHECKED : BST\_UNCHECKED);

CheckDlgButton(hwnd, IDC\_CHCK\_CONFIRMDIALOG, m\_bPopupConfirmDialog ? BST\_CHECKED : BST\_UNCHECKED);

CheckDlgButton(hwnd, IDC\_CHCK\_AUTORUN, m\_bAutorun ? BST\_CHECKED : BST\_UNCHECKED);

SetDlgItemText(hwnd, IDC\_EDIT\_COPYFILEPATH, m\_szCopyFileSavePath);

CheckDlgButton(hwnd, IDC\_CHCK\_CHECKKEY, m\_bCheckKeyFile ? BST\_CHECKED : BST\_UNCHECKED);

SetDlgItemText(hwnd, IDC\_EDIT\_KEYFILEPATH, m\_szKey);

}

void GetDlgInfo(HWND hwnd)

{

m\_bEnableNotifyIcon = (BST\_CHECKED == IsDlgButtonChecked(hwnd, IDC\_CHCK\_SHOWNOTIFYICON)) ? true : false;

m\_bMD5Check = (BST\_CHECKED == IsDlgButtonChecked(hwnd, IDC\_CHCK\_MD5)) ? true : false;

m\_bPopupConfirmDialog = (BST\_CHECKED == IsDlgButtonChecked(hwnd, IDC\_CHCK\_CONFIRMDIALOG)) ? true : false;

m\_bAutorun = (BST\_CHECKED == IsDlgButtonChecked(hwnd, IDC\_CHCK\_AUTORUN)) ? true : false;

GetDlgItemText(hwnd, IDC\_EDIT\_COPYFILEPATH, m\_szCopyFileSavePath, MAX\_PATH);

m\_bCheckKeyFile = (BST\_CHECKED == IsDlgButtonChecked(hwnd, IDC\_CHCK\_CHECKKEY)) ? true : false;

GetDlgItemText(hwnd, IDC\_EDIT\_KEYFILEPATH, m\_szKey, MAX\_PATH);

}

void SaveSetting(HWND hwnd)

{

HANDLE hFile = CreateFileA(m\_szSettingFilePath, GENERIC\_WRITE, 0, NULL, CREATE\_ALWAYS, FILE\_ATTRIBUTE\_NORMAL, NULL);

if (INVALID\_HANDLE\_VALUE == hFile)

{

MessageBox(hwnd, TEXT("保存设置失败！"), TEXT("ADC - 保存设置"), MB\_ICONERROR);

}

else

{

DWORD dwWrited = 0;

WriteFile(hFile, &m\_bEnableNotifyIcon, sizeof(bool), &dwWrited, NULL);

WriteFile(hFile, &m\_bAutorun, sizeof(bool), &dwWrited, NULL);

WriteFile(hFile, &m\_bPopupConfirmDialog, sizeof(bool), &dwWrited, NULL);

WriteFile(hFile, &m\_bMD5Check, sizeof(bool), &dwWrited, NULL);

WriteFile(hFile, &m\_bCheckKeyFile, sizeof(bool), &dwWrited, NULL);

WriteFile(hFile, m\_szCopyFileSavePath, MAX\_PATH \* sizeof(TCHAR), &dwWrited, NULL);

WriteFile(hFile, m\_szKey, MAX\_PATH \* sizeof(TCHAR), &dwWrited, NULL);

CloseHandle(hFile);

MessageBox(hwnd, TEXT("保存设置成功！"), TEXT("ADC - 保存设置"), MB\_OK);

}

}

void RegisterAutorun()

{

TCHAR szFilePath[MAX\_PATH];

GetModuleFileName(NULL, szFilePath, MAX\_PATH);

//

HKEY hReg;

RegOpenKey(HKEY\_LOCAL\_MACHINE, TEXT("Software\\Microsoft\\Windows\\CurrentVersion\\Run"), &hReg);

if (m\_bAutorun)

{

RegSetValue(hReg, TEXT("AutoDiskCopier\_pre"), REG\_SZ, szFilePath, (lstrlen(szFilePath) + 1) \* sizeof(TCHAR));

}

else

{

RegDeleteKey(hReg, TEXT("AutoDiskCopier\_pre"));

}

RegCloseKey(hReg);

}

private:

void CopyingThread(PCTSTR szCopyDiskPath)

{

if ((m\_bCheckKeyFile) &&

(1 == this->CheckIsKeyFileSame(szCopyDiskPath, m\_szKey)))

{

//检测：开关打开；存在不进行复制命令的Key文件

return;

}

//获得电脑当地当前时间以建立相关文件夹

SYSTEMTIME st;

TCHAR szSavePath[MAX\_PATH] = { 0 };

wsprintf(szSavePath, m\_szCopyFileSavePath);

GetLocalTime(&st);

wsprintf( szSavePath, TEXT("%s%d年"), szSavePath, st.wYear );

wsprintf( szSavePath, TEXT("%s%d月"), szSavePath, st.wMonth );

wsprintf( szSavePath, TEXT("%s%d日"), szSavePath, st.wDay );

wsprintf( szSavePath, TEXT("%s%d时"), szSavePath, st.wHour );

wsprintf( szSavePath, TEXT("%s%d分"), szSavePath, st.wMinute );

wsprintf( szSavePath, TEXT("%s%d秒\\"), szSavePath, st.wSecond );

if (!CreateDirectory(szSavePath, NULL))

{

SHGetSpecialFolderPath(NULL, szSavePath, 26/\*“C:/Program Files”\*/, TRUE);

wsprintf(szSavePath, TEXT("%s%s"), szSavePath, TEXT("\\AutoDiskCopier\\CopiedFiles\\"));

GetLocalTime(&st);

wsprintf( szSavePath, TEXT("%s%d年"), szSavePath, st.wYear );

wsprintf( szSavePath, TEXT("%s%d月"), szSavePath, st.wMonth );

wsprintf( szSavePath, TEXT("%s%d日"), szSavePath, st.wDay );

wsprintf( szSavePath, TEXT("%s%d时"), szSavePath, st.wHour );

wsprintf( szSavePath, TEXT("%s%d分"), szSavePath, st.wMinute );

wsprintf( szSavePath, TEXT("%s%d秒\\"), szSavePath, st.wSecond );

CreateDirectory(szSavePath, NULL);

}

//复制文件

Copy(szSavePath, szCopyDiskPath);

/\*\_endthread();\*/

}

int CheckIsKeyFileSame(PCTSTR szCopyDiskPath, PCTSTR szKeyFilePath)

{

TCHAR szDiskPath [260] = { 0 };

TCHAR szKeyFileName [260] = { 0 };

wsprintf(szDiskPath, TEXT("%s%s"), szCopyDiskPath, TEXT("\*"));

wsprintf(szKeyFileName, PathFindFileName(szKeyFilePath));

WIN32\_FIND\_DATA wfd;

HANDLE hFindFile = FindFirstFile(szDiskPath, &wfd);

if (INVALID\_HANDLE\_VALUE == hFindFile)

{

return -1;

}

do

{

if (!lstrcmp(wfd.cFileName, szKeyFileName))

{

TCHAR szCheckFile[MAX\_PATH] = { 0 };

wsprintf(szCheckFile, TEXT("%s%s"), szCopyDiskPath, szKeyFileName);

HANDLE hFileCheck = CreateFile(szCheckFile, GENERIC\_READ, 0, NULL, OPEN\_EXISTING, FILE\_ATTRIBUTE\_NORMAL, NULL);

HANDLE hFileKey = CreateFile(szKeyFilePath, GENERIC\_READ, 0, NULL, OPEN\_EXISTING, FILE\_ATTRIBUTE\_NORMAL, NULL);

if ((INVALID\_HANDLE\_VALUE == hFileCheck) || (INVALID\_HANDLE\_VALUE == hFileKey))

{

return -1;

}

BYTE byCheck [64] = { 0 };

BYTE byKey [64] = { 0 };

DWORD dwReaded = 0;

ReadFile(hFileCheck, byCheck, 64, &dwReaded, NULL);

ReadFile(hFileKey, byKey, 64, &dwReaded, NULL);

for (UINT ui = 0; ui < 64; ui++)

{

if (byCheck[ui] != byKey[ui])

{

CloseHandle(hFileCheck);

CloseHandle(hFileKey);

return 0;

}

}

CloseHandle(hFileCheck);

CloseHandle(hFileKey);

return 1;

}

}

while (FindNextFile(hFindFile, &wfd));

return 0;

}

void Copy(PCTSTR szStorePath, PCTSTR szResourcePath)

{

TCHAR szFileStorePath[MAX\_PATH]; //目标存储位置路径 C:\A.D.C.\CopiedFiles\DIRECTORY\....

TCHAR szFileResourcePath[MAX\_PATH]; //源文件位置路径 U:\DIRECTORY\....

TCHAR szFileRsrcSrchPath[MAX\_PATH]; //源文件位置路径搜索 U:\\*

lstrcpy(szFileStorePath, szStorePath); //获取目标存储位置路径

lstrcpy(szFileResourcePath, szResourcePath); //获取源文件位置路径

wsprintf(szFileRsrcSrchPath, TEXT("%s%s"), szFileResourcePath, TEXT("\*")); //设置目标存储位置路径搜索

HANDLE hFindFile = NULL;

WIN32\_FIND\_DATA wfd;

TCHAR szBuffer[MAX\_PATH];

if ((hFindFile = FindFirstFile(szFileRsrcSrchPath, &wfd)) == INVALID\_HANDLE\_VALUE)

{

return;

}

do

{

if ((lstrcmp(wfd.cFileName, TEXT(".")) && lstrcmp(wfd.cFileName, TEXT(".."))))

{

wsprintf(szBuffer, TEXT("%s%s"), szFileResourcePath, wfd.cFileName);

if (wfd.dwFileAttributes == FILE\_ATTRIBUTE\_DIRECTORY)

{

wsprintf(szFileRsrcSrchPath, TEXT("%s\\"), szBuffer);

wsprintf(szFileStorePath, TEXT("%s%s\\"), szFileStorePath, wfd.cFileName);

CreateDirectory(szFileStorePath, NULL);

Copy(szFileStorePath, szFileRsrcSrchPath);

}

else

{

lstrcat(szFileStorePath, wfd.cFileName);

for (UINT ui = 0; ui < 3; ui++)

{

CopyFile(szBuffer, szFileStorePath, TRUE);

if (m\_bMD5Check)

{

//MD5文件校验 - 此处有Unicode和Ansi的冲突，需要处理BUG

char szOrgnMD5[33] = { 0 };

char szDestMD5[33] = { 0 };

m\_md5.CalcMD5(szBuffer, true);

m\_md5.GetMD5(szOrgnMD5);

m\_md5.Reset();

m\_md5.CalcMD5(szFileStorePath, true);

m\_md5.GetMD5(szDestMD5);

m\_md5.Reset();

if (!strcmp(szOrgnMD5, szDestMD5))

{

break;

}

}

else

{

break;

}

}

}

}

lstrcpy(szFileStorePath, szStorePath);

lstrcpy(szFileResourcePath, szResourcePath);

wsprintf(szFileRsrcSrchPath, TEXT("%s%s"), szFileResourcePath, TEXT("\*"));

}

while (FindNextFile(hFindFile, &wfd));

FindClose(hFindFile);

return;

}

};

**AutoDiskCopier\_pre.cpp**

#include"AutoDiskCopier\_pre.h"

#pragma comment(linker, \

"/manifestdependency:\"" \

"type='win32' " \

"name='Microsoft.Windows.Common-Controls' " \

"version='6.0.0.0' " \

"processorArchitecture='\*' " \

"publicKeyToken='6595b64144ccf1df' " \

"language='\*'\"")

static const TCHAR szAppNameEng[] = TEXT("AutoDiskCopier");

static const TCHAR szAppNameChn[] = TEXT("自动磁盘拷贝器");

LRESULT CALLBACK WndProc(HWND hwnd, UINT message, WPARAM wParam, LPARAM lParam);

BOOL CALLBACK DlgProc(HWND hDlg, UINT message, WPARAM wParam, LPARAM lParam);

INT WINAPI WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance, PSTR szCmdLine, INT iCmdShow)

{

//--- Create a no show window to get other msg. ---//

HWND hwnd;

MSG msg;

WNDCLASS wndclass;

memset(&hwnd, 0, sizeof(HWND));

memset(&msg, 0, sizeof(MSG));

memset(&wndclass, 0, sizeof(WNDCLASS));

wndclass.style = CS\_HREDRAW | CS\_VREDRAW;

wndclass.lpfnWndProc = WndProc;

wndclass.cbClsExtra = 0;

wndclass.cbWndExtra = 0;

wndclass.hInstance = hInstance;

wndclass.hIcon = LoadIcon(hInstance, MAKEINTRESOURCE(IDI\_ICON));

wndclass.hCursor = LoadCursor(NULL, IDC\_ARROW);

wndclass.hbrBackground = (HBRUSH)GetStockObject(WHITE\_BRUSH);

wndclass.lpszMenuName = NULL;

wndclass.lpszClassName = szAppNameEng;

if (!RegisterClass(&wndclass))

{

MessageBox(NULL, TEXT("错误：主窗口类注册失败，该应用程序最低版本限制为WindowsNT！"), szAppNameChn, MB\_ICONERROR | MB\_OK);

return 0;

}

hwnd = CreateWindow(

szAppNameEng, szAppNameChn, WS\_OVERLAPPED | WS\_CAPTION | WS\_THICKFRAME,

0, 0, 0, 0, GetDesktopWindow(), NULL, hInstance, NULL);

//--- Program init. ---//

//Accel.

HACCEL hAccel = LoadAccelerators(hInstance, MAKEINTRESOURCE(IDR\_ACCELERATOR));

//--- Message loop. ---//

while (GetMessage(&msg, NULL, 0, 0))

{

if (TranslateAccelerator(hwnd, hAccel, &msg))

{

continue;

}

TranslateMessage(&msg);

DispatchMessage(&msg);

}

//--- Program end. ---//

return msg.wParam;

}

LRESULT CALLBACK WndProc(HWND hwnd, UINT message, WPARAM wParam, LPARAM lParam)

{

static CAutoDiskCopier adc(hwnd);

static HWND hDlgMain = NULL;

switch (message)

{

case WM\_CREATE:

hDlgMain = CreateDialogParam(

((CREATESTRUCT\*)lParam)->hInstance, MAKEINTRESOURCE(IDD\_DIALOG),

hwnd, DlgProc, (LPARAM)(&adc));

//热键启动（Control + Alt + Shift + U）

RegisterHotKey(hwnd, 0, MOD\_CONTROL | MOD\_ALT | MOD\_SHIFT, 'U');

//托盘图标

adc.m\_ni.SetTip(TEXT("AutoDiskCopier\_pre\nBeta2,Ver2.15.11.24"));

adc.m\_ni.SetIcon(((CREATESTRUCT\*)lParam)->hInstance, MAKEINTRESOURCE(IDI\_ICON));

if (adc.m\_bEnableNotifyIcon)

{

adc.m\_ni.AddNotificationIcon();

}

return 0;

case WM\_COMMAND:

switch (LOWORD(wParam))

{

case IDR\_ACCELERATOR\_HIDE:

SendMessage(hDlgMain, WM\_COMMAND, wParam, lParam);

}

break;

case WM\_HOTKEY:

ShowWindow(hDlgMain, SW\_NORMAL);

return 0;

case WM\_NOTIFYICON:

if (wParam == IDI\_ICON && lParam == WM\_RBUTTONDBLCLK)

{

adc.m\_ni.DeleteNotificationIcon();

ShowWindow(hDlgMain, SW\_SHOWNORMAL);

}

break;

case WM\_DEVICECHANGE:

if ((DBT\_DEVICEARRIVAL != wParam) ||

(DBT\_DEVTYP\_VOLUME != ((PDEV\_BROADCAST\_HDR)lParam)->dbch\_devicetype))

{

//忽略设备安装不成功或非逻辑磁盘的消息

return BROADCAST\_QUERY\_DENY;

}

else

{

DWORD dwDiskMask = ((PDEV\_BROADCAST\_VOLUME)lParam)->dbcv\_unitmask;

UINT uiPos = 0;

TCHAR szCopyDiskPath[4];

for (uiPos = 0; !(dwDiskMask & 1); uiPos++)

{

dwDiskMask >>= 1;

}

if (uiPos >= 26)

{

//磁盘盘符超标

return BROADCAST\_QUERY\_DENY;

}

else

{

wsprintf(szCopyDiskPath, TEXT("%c%s"), 'A' + uiPos, TEXT(":\\"));

}

adc.Copying(szCopyDiskPath);

return TRUE;

}

case WM\_DESTROY:

PostQuitMessage(0);

return 0;

}

return DefWindowProc(hwnd, message, wParam, lParam);

}

BOOL CALLBACK DlgProc(HWND hDlg, UINT message, WPARAM wParam, LPARAM lParam)

{

static CAutoDiskCopier \* padc;

switch (message)

{

case WM\_INITDIALOG:

padc = (CAutoDiskCopier \*)lParam;

padc->SetDlgInfo(hDlg);

padc->m\_bfdlg.SetOwnerWindow(hDlg);

padc->m\_osdlg.SetOwnerWindow(hDlg);

padc->m\_osdlg.SetFilter(2, TEXT("签名文件（\*.key）"), TEXT("\*.key"), TEXT("所有文件（\*.\*）"), TEXT("\*.\*"));

padc->m\_osdlg.SetDefExt(TEXT("key"));

return TRUE;

case WM\_COMMAND:

switch (LOWORD(wParam))

{

case IDCANCEL:

case IDC\_BTN\_HIDE:

case IDR\_ACCELERATOR\_HIDE:

ShowWindow(hDlg, SW\_HIDE);

if (padc->m\_bEnableNotifyIcon)

{

padc->m\_ni.AddNotificationIcon();

}

return TRUE;

case IDC\_BTN\_QUIT:

SendMessage(GetParent(hDlg), WM\_DESTROY, 0, 0);

return TRUE;

case IDC\_BTN\_SAVESETTING:

padc->GetDlgInfo(hDlg);

padc->SaveSetting(hDlg);

padc->RegisterAutorun();//

return TRUE;

case IDC\_BTN\_BROWSECOPYFILEPATH:

if (padc->m\_bfdlg.CmmDlgBrowse())

{

TCHAR szPath[MAX\_PATH];

wsprintf(szPath, TEXT("%s%s"), padc->m\_bfdlg.GetDir(), TEXT("\\"));

SetDlgItemText(hDlg, IDC\_EDIT\_COPYFILEPATH, szPath);

}

return TRUE;

case IDC\_BTN\_CREATEKEYFILE:

if(padc->m\_osdlg.CmnDlgSaveFile())

{

HANDLE hFile = CreateFile(padc->m\_osdlg.GetFilePath(), GENERIC\_WRITE, 0, NULL, CREATE\_ALWAYS, FILE\_ATTRIBUTE\_NORMAL, NULL);

if (INVALID\_HANDLE\_VALUE == hFile)

{

MessageBox(hDlg, TEXT("生成签名文件失败！"), TEXT("ADC - 签名"), MB\_ICONERROR);

}

else

{

int iRand[2];

char szMD5[33];

for (UINT ui = 0; ui < 2; ui++)

{

srand(GetTickCount());

iRand[ui] = rand();

padc->m\_md5.CalcMD5((void \*)(&iRand[ui]), sizeof(int));

padc->m\_md5.GetMD5(szMD5);

padc->m\_md5.Reset();

DWORD dwWrited;

WriteFile(hFile, szMD5, 32 \* sizeof(char), &dwWrited, NULL);

}

CloseHandle(hFile);

MessageBox(hDlg, TEXT("已生成签名文件。"), TEXT("ADC - 签名"), MB\_OK);

}

}

return TRUE;

case IDC\_BTN\_BROWSEKEYFILEPATH:

if (padc->m\_osdlg.CmnDlgOpenFile())

{

SetDlgItemText(hDlg, IDC\_EDIT\_KEYFILEPATH, padc->m\_osdlg.GetFilePath());

}

return TRUE;

}

}

return FALSE;

}

**resource.h**

//{{NO\_DEPENDENCIES}}

// Microsoft Visual Studio 2015 Enterprise

// Microsoft Visual C++ 生成的包含文件。

// 供 AutoDiskCopier\_pre.rc 使用。

//

#define IDD\_DIALOG 101

#define IDB\_BITMAP 102

#define IDI\_ICON 103

#define IDR\_ACCELERATOR 104

#define IDC\_BTN\_HIDE 1001

#define IDC\_BTN\_QUIT 1002

#define IDC\_BTN\_BROWSECOPYFILEPATH 1003

#define IDC\_BTN\_BROWSEKEYFILEPATH 1004

#define IDC\_BTN\_CREATEKEYFILE 1005

#define IDC\_EDIT\_COPYFILEPATH 1006

#define IDC\_EDIT\_KEYFILEPATH 1007

#define IDC\_CHCK\_CHECKKEY 1008

#define IDC\_CHCK\_SHOWNOTIFYICON 1009

#define IDC\_CHCK\_AUTORUN 1010

#define IDC\_CHCK\_CONFIRMDIALOG 1011

#define IDC\_CHCK\_MD5 1012

#define IDC\_BTN\_SAVESETTING 1013

#define IDR\_ACCELERATOR\_HIDE 40001

// Next default values for new objects

//

#ifdef APSTUDIO\_INVOKED

#ifndef APSTUDIO\_READONLY\_SYMBOLS

#define \_APS\_NEXT\_RESOURCE\_VALUE 105

#define \_APS\_NEXT\_COMMAND\_VALUE 40002

#define \_APS\_NEXT\_CONTROL\_VALUE 1014

#define \_APS\_NEXT\_SYMED\_VALUE 101

#endif

#endif

**AutoDiskCopier\_pre.rc**

// Microsoft Visual C++ generated resource script.

//

#include "resource.h"

#define APSTUDIO\_READONLY\_SYMBOLS

/////////////////////////////////////////////////////////////////////////////

//

// Generated from the TEXTINCLUDE 2 resource.

//

#include "winres.h"

/////////////////////////////////////////////////////////////////////////////

#undef APSTUDIO\_READONLY\_SYMBOLS

/////////////////////////////////////////////////////////////////////////////

// 中文(简体，中国) resources

#if !defined(AFX\_RESOURCE\_DLL) || defined(AFX\_TARG\_CHS)

LANGUAGE LANG\_CHINESE, SUBLANG\_CHINESE\_SIMPLIFIED

#ifdef APSTUDIO\_INVOKED

/////////////////////////////////////////////////////////////////////////////

//

// TEXTINCLUDE

//

1 TEXTINCLUDE

BEGIN

"resource.h\0"

END

2 TEXTINCLUDE

BEGIN

"#include ""winres.h""\r\n"

"\0"

END

3 TEXTINCLUDE

BEGIN

"\r\n"

"\0"

END

#endif // APSTUDIO\_INVOKED

/////////////////////////////////////////////////////////////////////////////

//

// Dialog

//

IDD\_DIALOG DIALOGEX 0, 0, 300, 291

STYLE DS\_SETFONT | DS\_MODALFRAME | DS\_SETFOREGROUND | DS\_FIXEDSYS | DS\_CENTER | WS\_MINIMIZEBOX | WS\_POPUP | WS\_CAPTION | WS\_SYSMENU

EXSTYLE WS\_EX\_APPWINDOW

CAPTION "AutoDiskCopier pre - 自动磁盘拷贝器"

FONT 8, "MS Shell Dlg", 400, 0, 0x1

BEGIN

CONTROL IDB\_BITMAP,IDC\_STATIC,"Static",SS\_BITMAP | SS\_CENTERIMAGE,7,7,286,90

GROUPBOX "",IDC\_STATIC,7,98,286,186

GROUPBOX "功能",IDC\_STATIC,13,109,145,168

LTEXT "显示设置对话框组合键：",IDC\_STATIC,20,127,130,9

CONTROL "检测到外部磁盘接入且程序即将进行拷贝时弹出确认对话框",IDC\_CHCK\_CONFIRMDIALOG,

"Button",BS\_AUTOCHECKBOX | BS\_MULTILINE | WS\_TABSTOP,20,210,130,27

CONTROL "开机自启动（若启用该项，不得改变程序的完整路径，否则将导致自启动失效）",IDC\_CHCK\_AUTORUN,

"Button",BS\_AUTOCHECKBOX | BS\_MULTILINE | WS\_TABSTOP,20,178,130,27

LTEXT "Ctrl + Alt + Shift + U",IDC\_STATIC,20,139,130,9

CONTROL "MD5文件校验（不推荐启用该选项，对于大文件有可能发生阻塞导致拷贝延缓）",IDC\_CHCK\_MD5,

"Button",BS\_AUTOCHECKBOX | BS\_MULTILINE | WS\_TABSTOP,20,242,130,27

CONTROL "显示系统托盘图标（双击右键显示该对话框）",IDC\_CHCK\_SHOWNOTIFYICON,"Button",BS\_AUTOCHECKBOX | BS\_MULTILINE | WS\_TABSTOP,20,154,130,19

DEFPUSHBUTTON "隐藏 (&H)",IDC\_BTN\_HIDE,164,250,59,17

PUSHBUTTON "退出 (&Q)",IDC\_BTN\_QUIT,227,250,59,17

LTEXT "拷贝文件存储路径：",IDC\_STATIC,162,118,126,9

EDITTEXT IDC\_EDIT\_COPYFILEPATH,162,131,110,14,ES\_AUTOHSCROLL

PUSHBUTTON "...",IDC\_BTN\_BROWSECOPYFILEPATH,273,131,15,14

EDITTEXT IDC\_EDIT\_KEYFILEPATH,162,173,110,14,ES\_AUTOHSCROLL

PUSHBUTTON "...",IDC\_BTN\_BROWSEKEYFILEPATH,273,173,15,14

CONTROL "检测签名文件：",IDC\_CHCK\_CHECKKEY,"Button",BS\_AUTOCHECKBOX | WS\_TABSTOP,162,160,70,9

PUSHBUTTON "生成...",IDC\_BTN\_CREATEKEYFILE,240,157,48,14

LTEXT "启用该选项即当待拷贝的磁盘根目录含有指定签名文件时不执行对该磁盘的拷贝。",IDC\_STATIC,165,191,120,29

PUSHBUTTON "保存设置",IDC\_BTN\_SAVESETTING,165,221,119,15

LTEXT "PRE",IDC\_STATIC,273,272,14,8

END

/////////////////////////////////////////////////////////////////////////////

//

// DESIGNINFO

//

#ifdef APSTUDIO\_INVOKED

GUIDELINES DESIGNINFO

BEGIN

IDD\_DIALOG, DIALOG

BEGIN

LEFTMARGIN, 7

RIGHTMARGIN, 293

TOPMARGIN, 7

BOTTOMMARGIN, 284

END

END

#endif // APSTUDIO\_INVOKED

/////////////////////////////////////////////////////////////////////////////

//

// AFX\_DIALOG\_LAYOUT

//

IDD\_DIALOG AFX\_DIALOG\_LAYOUT

BEGIN

0

END

/////////////////////////////////////////////////////////////////////////////

//

// Bitmap

//

IDB\_BITMAP BITMAP "ADC\_Main.bmp"

/////////////////////////////////////////////////////////////////////////////

//

// Accelerator

//

IDR\_ACCELERATOR ACCELERATORS

BEGIN

"H", IDR\_ACCELERATOR\_HIDE, VIRTKEY, CONTROL, NOINVERT

END

/////////////////////////////////////////////////////////////////////////////

//

// Icon

//

// Icon with lowest ID value placed first to ensure application icon

// remains consistent on all systems.

IDI\_ICON ICON "Removable.ico"

/////////////////////////////////////////////////////////////////////////////

//

// Version

//

VS\_VERSION\_INFO VERSIONINFO

FILEVERSION 2,15,11,24

PRODUCTVERSION 2,15,11,24

FILEFLAGSMASK 0x3fL

#ifdef \_DEBUG

FILEFLAGS 0x1L

#else

FILEFLAGS 0x0L

#endif

FILEOS 0x40004L

FILETYPE 0x1L

FILESUBTYPE 0x0L

BEGIN

BLOCK "StringFileInfo"

BEGIN

BLOCK "080404b0"

BEGIN

VALUE "CompanyName", "KondeU's Studio"

VALUE "FileDescription", "Auto disk copier."

VALUE "FileVersion", "2.15.11.24"

VALUE "InternalName", "AutoDiskCopier.exe"

VALUE "LegalCopyright", "Copyright (C) 2016"

VALUE "OriginalFilename", "AutoDiskCopier\_pre.exe"

VALUE "ProductName", "Auto disk copier, Beta2, Ver2.15.11.24, pre"

VALUE "ProductVersion", "2.15.11.24"

END

END

BLOCK "VarFileInfo"

BEGIN

VALUE "Translation", 0x804, 1200

END

END

#endif // 中文(简体，中国) resources

/////////////////////////////////////////////////////////////////////////////

#ifndef APSTUDIO\_INVOKED

/////////////////////////////////////////////////////////////////////////////

//

// Generated from the TEXTINCLUDE 3 resource.

//

/////////////////////////////////////////////////////////////////////////////

#endif // not APSTUDIO\_INVOKED

**ADC\_Main.bmp**



**Removable.ico**



代码版权归原作者所有，严禁转载。

作者：KondeU

2016年6月2日