# ****CppFastSolution****

****（CFS)****

***PefileParsing:***

此类可实现解析PE文件并添加一个新节区的功能（此类仅windows），

也可用来解析PE文件

explicit PefileParsing

(

\_In\_ string filepath,

\_In\_ string savepath

)：

类构造方法

参数：

filepath:用于传入要加壳或是要新加区段的文件路径

savepath:新区段添加完成后要保存新文件的路径

返回值：无

void setFilePath

(

\_In\_ string filepath,

\_In\_ string savepath

)：

用于单独设置新加区段文件与完成后保存文件路径方法

参数：

filepath:用于传入要加壳或是要新加区段的文件路径

savepath:新区段添加完成后要保存新文件的路径

返回值：无

BYTE\* Add\_NewSection

(

\_Inout\_ BYTE\* OldData,

\_Inout\_ SIZE\_T &fsize,

\_Inout\_ int& back,

\_In\_ const BYTE\* Section\_data,

\_In\_ const BYTE\* Section\_name,

\_In\_ const SIZE\_T Section\_size);

用于添加新区段的方法

参数：

OldData：原有要新加区段的文件数据，从GetFilePE函数中取出的数据

fsize：OldData缓冲区大小，原有文件数据大小

back：错误值详细信息

Section\_data：要新加的区段数据

Section\_name：新加节区表名称

Section\_size：新加节区数据大小

返回值：

调用成功返回新加区段后的文件数据，可将此函数返回值直接保存至文件，

保存后的文件就是新加节区后的PE文件

调用失败返回NULL,错误详细信息记录在back参数中

void ClearPEBuff(\_In\_ BYTE\* File\_Data);

清理内存中已保存的文件数据，防止内存泄漏，用于释放Add\_NewSection返回值，在Add\_NewSection返回值已经被保存至文件后

参数：

File\_Data：要被释放的文件数据缓冲区指针

返回值：无

static PefileParsing \*getInterface (

\_In\_ string filepath,

\_In\_ string savepath);

此类提供的一个单例模式访问方法

参数：

filepath：要新加节区的文件路径，用于初始化类的构造函数

savepath：新加节区后的PE文件保存路径，用于初始化构造函数

返回值：

一个全局静态的单例PefileParsing对象

static int GetPEfileType

(

\_In\_ BYTE\* FileData,

\_Inout\_ int& back);

用于获取PE文件类型

参数：

FileData：PE文件读入内存后的缓冲区数据指针

back：错误详细信息

返回值：

成功返回PE文件类型，失败返回\_\_ERROR，错误详细信息记录于back

static PIMAGE\_DOS\_HEADER GetFile\_Dosheader

(

\_In\_ BYTE\* File\_Data,

\_Inout\_ int& back);

用于获取PE文件DOS头

参数：

File\_Data：读入内存的PE文件数据

back：错误值详细信息

返回值：

成功返回PIMAGE\_DOS\_HEADER结构体指针，失败返回空

static PIMAGE\_NT\_HEADERS GetFile\_NTheader

(

\_In\_ BYTE\* File\_Data,

\_Inout\_ int& back);

用于获取PE（NT）头

参数：

File\_Data：读入内存的PE文件数据

back：错误值详细信息

返回值：

成功返回PIMAGE\_NT\_HEADERS结构体，失败返回NULL

static PIMAGE\_FILE\_HEADER GetFile\_header

(

\_In\_ BYTE\* File\_Data,

\_Inout\_ int& back);

用于获取PE文件头

参数：

File\_Data：读入内存的PE文件数据

back：错误值详细信息

返回值：

成功返回PIMAGE\_FILE\_HEADER结构体，失败返回NULL

static PIMAGE\_OPTIONAL\_HEADER GetFile\_Optheader

(

\_In\_ BYTE\* File\_Data,

\_Inout\_ int& back);

用于获取PE文件扩展头

参数：

File\_Data：读入内存的PE文件数据

back：错误值详细信息

返回值：

成功返回PIMAGE\_OPTIONAL\_HEADER结构体，失败返回NULL

static PIMAGE\_SECTION\_HEADER GetLastSection

(

\_In\_ BYTE\* File\_Data,

\_Inout\_ int& back);

用于获取PE文件最后一个节区表后的空白部分，

并将其格式化为一个PIMAGE\_SECTION\_HEADER结构体

参数：

File\_Data：读入内存的PE文件数据

back：错误详细信息

返回值：

成功返回一个PIMAGE\_SECTION\_HEADER结构体，失败返回NULL

static SIZE\_T Section\_Alignment

(

\_In\_ SIZE\_T File\_Size,

\_In\_ SIZE\_T Alignment);

用于内存对齐

参数：

File\_Siz：数据实际大小

Alignment：内存粒度，应该是第一个参数的倍数

返回值：

数据实际大小，经内存粒度对齐后的内存占用大小

int SaveFile

(

\_In\_ BYTE\*FileData,

\_In\_ const DWORD FileSize);

用于保存添加新区段后内存数据为一个新PE文件，调用完此函数后记得手动释放FileData

参数：

FileData：添加过新区段后的文件数据指针，通常是Add\_NewSection的返回值

FileSize：FileData的大小

返回值：

成功返回\_\_SUCCESS，失败返回错误详细信息

BYTE\* GetFilePE

(

\_Inout\_ DWORD& back,

\_Inout\_ SIZE\_T& fsize);

用于打开获取要新加节区文件（要被加壳文件）的原有数据，以供其他函数调用

参数：

back,：用于记录详细错误值

fsize：用于返回文件大小

返回值：

成功·返回读到的文件数据缓冲区指针，失败返回NULL

static PIMAGE\_SECTION\_HEADER GetSectionForName

(

\_In\_ BYTE\* FileData,

\_In\_ string secName,

\_Inout\_ int& backNum

);

根据节区名获取指定的节区头

参数：

BYTE\* FileData：读取到的文件数据此数据可通过 GetFilePE函数获取

string secName：指定要获取节区头的节区名

int& backNum：错误值详细信息

返回值：

调用成功则返回一个PIMAGE\_SECTION\_HEADER指针，失败返回NULL，并且会将错误原因写入

backNum反馈，如果此函数返回值为NULL，而且backNum值也为\_\_SUCCESS，那说明，指定

的节区不存在，所以判断此函数是否正确执行的标准应该是判断backNum的值是否为\_\_SUCCESS

static BYTE\* GetSectionData

(

\_In\_ BYTE\* FileData,

\_In\_ string secName,

\_Inout\_ unsigned long& Secsize,

\_In\_ int& back

);

根据节区名获取到具体的节区数据

参数：

BYTE\* FileData：读取到的文件数据，此数据可通过 GetFilePE函数获取

string secName：指定要获取的节区名

unsigned long& Secsize：通过此参数反馈给调用者的节区数据大小

int& back：详细错误信息

返回值：

执行成功返回一个BYTE指针，此指针指向读到的节区数据，执行失败返回NULL，

错误详细信息在back中。s

***TCPSocket*：**

此类实现一个基于TCP协议的socket套接字

explicit TCPSocket

(

\_In\_ int port,

\_In\_ unsigned long ip,

\_Inout\_ DWORD& back);

类构造函数，用于初始化套接字

参数：

port：通信端口

ip：ip设置，此重载方法主要针对INADDR\_ANY用于绑定本机地址，所以此值必须为它

back：详细错误信息

返回值：

无

explicit TCPSocket

(

\_In\_ int port,

\_In\_ string ip,

\_Inout\_ unsigned long& back);

类构造函数，用于初始化套接字

参数：

Port:通信端口

Ip:ip设置，此重载方法主要针对主动发起connect，

所以此参数是一个点分的ip地址字符串

Back:详细错误信息

返回值：

无

void setPort(\_In\_ int Port);

void setIp(\_In\_ unsigned long ip);

void setIp(\_In\_ string ip);

三个set方法，用于单独设置端口，以及ip地址，调用无参构造函数后需调用

参数：

Port:端口

Unsigned long ip:此值应该为INADDR\_ANY

String ip:此值为一个点分的ip地址字符串

返回值：

无

int SocketInit();

用于初始化套接字，使用无参构造函数，并且使用set方法初始

化各项必需属性后可使用此方法初始化套接字，

此方法使用一个用点分ip地址字符串初始化套接字，

故无法使用INADDR\_ANY

参数：无

返回值：成功返回\_\_SUCCESS,失败返回失败原因，一个非\_\_SUCCESS值

static TCPSocket\* getInterface

(

\_In\_ int port,

\_In\_ string ip,

\_Inout\_ unsigned long& back

);

类的单例模式访问接口

参数：

Port:通信端口

Ip：ip地址，此重载函数主要初始化一个点分的ip地址字符串

Back：相信错误信息

返回值：

一个单例类对象

static TCPSocket\* getInterface

(

\_In\_ int port,

\_In\_ unsigned long ip,

\_Inout\_ unsigned long& back

);

类的单例模式访问接口

参数：

Port:通信端口

Ip:ip地址，此重载函数主要初始化本机绑定ip，此值应为INADDR\_ANY

Back:详细错误信息

返回值：

一个单例类对象

int TCPbind();

服务端监听绑定函数

参数：无

返回值：

成功返回\_\_SUCCESS，失败返回\_\_ERROR

int TCPListen

(

\_In\_ void(\*callback)(SOCKET\*, sockaddr\_in\*),

\_In\_ int ListenNum

);

服务端监听函数,此函数使用回调机制处理已建立连接的客户端

套接字，在回调函数处理通信事件，回调函数中无需关闭参数传入的套接字，

此函数中会自动关闭已经结束通信客户端套接字，只需在服务端通信完

毕后调用对象的Close函数关闭服务端套接字即可,

注意;

此函数传入的回调函数不能是类成员函数，可以是静态函数和普通函数

参数：

(\*callback)(SOCKET\*, sockaddr\_in\*):

函数指针用于传入回调函数，回调函数有两个参数一个传入客户端套接字指针

一个sockaddr\_in结构体指针，用于记录已连接客户端其他详细信息，只能是

普通函数或是静态函数，不能是类成员函数

ListenNum:最大连接数,windows下此值不得大于128

返回值：

成功返回\_\_SUCCESS，失败返回失败原因

int TCPConnect

(

\_In\_ void (\*callback)(SOCKET,sockaddr\_in),

\_In\_ int& back

);

客户端主动建立连接函数,此函数同监听函数，也采用回调机制处理通信

事件，此函数一般用于主动发起连接，在客户端使用，与服务端通信结束后

调用对象成员函数Close关闭套接字即可

注意;

此函数传入的回调函数不能是类成员函数，可以是静态函数和普通函数

参数：

(\*callback)(SOCKET,sockaddr\_in)：

函数指针，用于传入回调函数，只能是普通函数或是静态函数，不能是类成员

函数

Back:详细错误信息，此值为WSAGetLastError（linux会有所不同）

函数获取到的失败原因

返回值：

成功返回\_\_SUCCESS，失败返回\_\_SOCK\_CONNECT\_ERROR，详细信息在back参数中

SOCKET\* TCPConnect

(

\_In\_ int& back

);

客户端主动建立连接函数,此函数同监听函数，此函数不采用回调机制，

直接返回一个连接成功后的套接字指针,此函数一般用于主动发起连接，

在客户端使用，与服务端通信结束后调用对象成员函数Close关闭套接字即可

参数：

Back：详细错误信息，此值为WSAGetLastError函数获取到的失败原因

返回值：

成功返回\_\_SUCCESS，失败返回NULL,详细信息在back参数中

int Close()

（linux：int TCPClose()）

关闭连接函数，通信完毕后，记得使用对象调用此函数关闭连接

参数：无

返回值：

成功返回\_\_SUCCESS,失败返回一个用WSAGetLastError（linux会有所不同）

获取到的错误详细信息

（注意：以下所有TCP与UDP的数据发送与接收函数都不会将缓冲区中的数据初始化为0

所以需要手动用memset将缓冲区中的所有数据都初始化为空，否则可能会出现乱码）

int TCPSend

(

\_In\_ SOCKET nsocket,（linux:\_In\_ int nsocket)

\_In\_ string buff,

\_In\_ unsigned long buffsize

);

数据发送函数，此函数非类成员函数，只是与此类配合使用，

一般放在连接建立后的回调函数中使用

参数：

Nsocket：已建立连接的套接字

Buff：要发送的数据缓冲区指针

Buffsize:数据缓冲区大小

返回值：

成功返回已发送的数据大小，失败返回-1

unsigned long TCPRecv

(

\_In\_ SOCKET nsocket,

\_In\_ char\* buff,

\_In\_ unsigned long buffsize

);

数据接收函数，此函数非类成员函数，同TCPSend

参数：

Nsocket:已建立连接的套接字

Buff:接收数据的缓冲区，接收到的数据保存在此

Buffsize:接收数据缓冲区大小

返回值：

成功返回已接收的数据大小，失败返回-1

unsigned long TCPSendRecv

(

\_In\_ SOCKET nsocket,（linux:\_In\_ int nsocket)

\_In\_ string sbuff,

\_In\_ unsigned long& sbuffsize,

\_Inout\_ char\* rbuff,

\_Inout\_ unsigned long& rbuffsize

);

数据发收函数，顾名思义，先发后收，可一次性完成一次收发任务

参数：

Nsocket:已建立连接的套接字

Sbuff:待发送数据的缓冲区指针

Sbuffsize:待发送数据字节大小，此值为引用调用成功后此值将被改变为已发送的数据大小

Rbuff:接收数据缓冲区，接收到的数据保存在此

Rbuffsize:接收缓冲区大小

返回值：

成功返回\_\_SUCCESS，失败返回WSAGetLastError（linux会有所不同）

返回的错误详细信息

unsigned long TCPRecvSend

(

\_In\_ SOCKET nsocket,（linux:\_In\_ int nsocket)

\_In\_ string sbuff,

\_In\_ unsigned long& sbuffsize,

\_Inout\_ char\* rbuff,

\_Inout\_ unsigned long& rbuffsize

)：

收发函数，顾名思义，先收后发，不同的是如果

没有收到数据就会直接返回\_\_SOCKET\_NO\_RECV，不会再发送数据

参数：

Nsocket:已建立连接的套接字

Sbuff:待发送数据的缓冲区指针

Sbuffsize:待发送数据字节大小，此值为引用调用成功后此值将被改变为已发送的数据大小

Rbuff:接收数据缓冲区，接收到的数据保存在此

Rbuffsize:接收缓冲区大小

返回值：

成功返回\_\_SUCCESS，失败返回WSAGetLastError（linux会有所不同）

错误详细信息，

如果发送不成功则返回\_\_SOCKET\_NO\_RECV，且不会将待发送数据发送

***UDPSocket：***

此类用于实现UDP通信

explicit UDPSocket

(

\_In\_ int port,

\_In\_ string ip,

\_Inout\_ unsigned long& back

);

类构造函数，用于初始化套接字

参数：

Port：端口设置

Ip：ip设置，此值为一个点分的Ip地址字符串

Back：详细错误信息

返回值：

无

explicit UDPSocket

(

\_In\_ int port,

\_In\_ unsigned long ip,

\_Inout\_ unsigned long& back

);

类构造函数，重载前一个函数，此重载函数针对INADDR\_ANY

参数：

Port：端口设置

Ip：ip设置，此值应该为INADDR\_ANY

Back：详细错误信息

返回值无

int SocketInit();

此方法介绍与TCPSocket完全一致

int UDPRecvfrom

(

\_Inout\_ char\* recvBuff,

\_Inout\_ unsigned long recvSize

);

UDP接收数据函数，需要手动将缓冲区中数据初始化为空，否则可能会出现乱码

参数：

recvBuff：存放接收数据的缓冲区指针

recvSize：缓冲区大小

返回值：

执行无误返回已接收的数据大小，失败返回-1(SOCKET\_ERROR)

int UDPSendto

(

\_In\_ const char\* sendBuff,

\_In\_ unsigned long sendSize

);

UDP发送数据函数,需要手动将缓冲区中数据初始化为空，否则可能会出现乱码

参数：

sendBuff：待发送数据缓冲区指针

sendSize：待发送数据大小

返回值：

执行无误返回已发送的数据大小，失败返回-1(SOCKET\_ERROR)

void setPort(\_In\_ int Port);

void setIp(\_In\_ string ip);

Set方法，在调用无参构造函数后应使用此三个方法初始化套接字必须属性

static UDPSocket\* getInterface

(

\_In\_ int port,

\_In\_ string ip,

\_Inout\_ unsigned long& back

);

类的单例模式访问接口

参数：

Port:通信端口

Ip：ip地址，此重载函数主要初始化一个点分的ip地址字符串

Back：相信错误信息

返回值：

一个单例类对象

static UDPSocket\* getInterface

(

\_In\_ int port,

\_In\_ unsigned long ip,

\_Inout\_ unsigned long& back

);

类的单例模式访问接口

参数：

Port:通信端口

Ip:ip地址，此重载函数主要初始化本机绑定ip，此值应为INADDR\_ANY

Back:详细错误信息

返回值：

一个单例类对象

int UDPbind();

本地绑定函数，用于服务端进行地址绑定

参数：无

返回值：成功返回\_\_SUCCESS，失败返回\_\_ERROR

int Close();

（linux：int UDPClose()）

套接字关闭函数，通信结束后使用此函数关闭套接字

参数：无

返回值：

成功返回\_\_SUCCESS,失败返回一个用WSAGetLastError（linux会有所不同）

获取到的错误详细信息

***Base64：***

Base64编码解码类，编解码base64码，此类所有函数均为静态函数，

故此类中的函数无需初始化即可使用

static string Base64Encode

(

\_In\_ string src,

\_In\_ u\_long srcsize

);

Base64编码函数，此函数对一个明文字符串进行base64编码

参数：

Src:需要编码的明文原字符串

Srcsize:原字符串的长度

返回值：

返回一个string对象，其值为一个base64编码字符串

static string Base64Decode

(

\_In\_ string src,

\_In\_ u\_long srcsize

);

Base64解码函数，此函数对一个已经过编码base64字符串进行解码

还原为原字符串

参数：

Src:需要进行解码的base64原字符串

Srcsize:原字符串长度

返回值：

此函数会判断密文长度，如果不是4的倍数则说明原字符串不是一个

Base64编码的字符串，直接返回空字符串，如果执行正确，返回一个

Base64字符串解码后的明文字符串对象

***SMTPSocket***

基于TCPSocket编写的的封装邮件发送类

explicit SMTPSocket();

类构造函数

无参数

无返回值

void setIpAddress(string ip);

set方法用于初始化SMTP服务器ip地址

无参数

无返回值

void setPort(int port);

用于初始化smtp服务器提供服务的端口，大多数为25

void setHostName(string hostname);

用于设置发送邮件主机名，此函数可不调用，发送邮件主机名有默认值

参数：

Hostname:本地主机名

返回值：无

void setUserPasswd

(

\_In\_ string user,

\_In\_ string passwd

);

用于初始化登录SMTP服务器的用户名与授权码，注意是授权码，不是邮箱登陆密码，当然

有些邮箱授权码就是邮箱密码

参数：

User:用户名

Passwd:授权码

返回值：

无

void setFromandTo

(

\_In\_ string from,

\_In\_ string sendto

);

用于设置发送地址与目的地址,此函数会将参数列表里的目的地址

也加入群发目的地址列表中

参数：

From:发送邮件的源地址

Sendto:接收邮件人

返回值:无

void setSubjectandMsg

(

\_In\_ string subject,

\_In\_ string Msg,

\_In\_ unsigned MsgSize

);

用于设置邮件标头与邮件内容

参数：

Subject:标题

Msg:邮件内容

MsgSize:内容长度

返回值：无

void addRcpto(string rcpto);

用于添加一个用户以供邮件群发使用，

此函数会将参数中的目的地址添加进群发邮件目的地址列表中，

可调用多次添加多个地址

参数：

Rcpto:一个邮件接收地址

返回值：无

int SMTPInit();

初始化用于smtp通信的网络套接字，此函数必须最先调用

int Login();

用于登录认证，在发送邮件主体前此函数必须先调用

int SendMail();

此函数只用于发送单个邮件,要先调用setFromandTo设置发出地址与发往地址

int SendMailAll();

此函数只用于群发邮件，群发邮件要先往群发邮件地址列表中添加地址，可先调用

setFromandTo设置邮件发出地址和首个发往地址，再多次调用addRcpto添加多个

发往地址

int SMTPQuit()

关闭smtp通信函数

static SMTPSocket\* getInterface();

单例模式访问接口

***MD5***

MD5编码处理类

#define SHORT\_MD5\_LEN 16

Short型md5值定长，调用者无需关心

#define CHAR\_MD5\_LEN 34

字符串型MD5值定长，输出的编码过后的MD5字符串一共32个字符，加上结束符\0一共33

再多预留一个字节，共34

static void MD5Encode

(

\_In\_ const char\* text,

\_In\_ size\_t len,

\_Inout\_ char\* dst

);

MD5编码函数，此函数用于进行MD5编码，一个静态函数，不进行实例化也可直接使用

参数：

const char\* text：待处理的原数据

size\_t len：原数据大小长度

char\* dst：生成后的MD5值，此参数为一个34字节定长的字符串指针

返回值：无

static bool MD5StrValidate

(

\_In\_ const char\* input1,

\_In\_ const char\* input2

);

MD5编码字符串校验比较函数，此函数用于校验两个MD5字符串是否一致

参数：

const char\* input1：MD5字符串1

const char\* input1：MD5字符串2

返回值：

两者一致返回true，否则返回false

***CEncryCore***

加密算法类核心基类，抽象类，此类大数处理部分基于GMP库

***RSA***

RSA加密类，此类可进行RSA加密解密操作，此类会生成两个1024位的随机数p、q,一固定值65537作为公钥，此类基于GMP库编写

void getKey

(

\_Inout\_ string& public\_key,

\_Inout\_ string& private\_key,

\_Inout\_ string& n

)

此函数用于生成公钥、私钥、与n值

参数：

\_Inout\_ string& public\_key：公钥，此参数用于返回公钥

\_Inout\_ string& private\_key：私钥，此参数用于返回私钥

\_Inout\_ string& n：此参数用于返回n值

返回值：无

string RSA\_Encode

(

\_In\_ const char\* IN\_Data,

\_Inout\_ size\_t& inoutLen,

\_In\_ string public\_key,

\_In\_ string n

)

此函数用于对数据进行加密，可输入一串数据，并输入其数据来对整串数据进行加密

参数：

\_In\_ const char\* IN\_Data：要被加密的明文

\_Inout\_ size\_t& inoutLen：明文长度，此值在函数执行完毕后变为加密后的密文长度

\_In\_ string public\_key：用于加密的公钥

\_In\_ string n：n值

返回值：

加密后的密文

string RSA\_Decode

(

\_In\_ string private\_key,

\_In\_ string n,

\_In\_ string c\_data,

\_Inout\_ size\_t& inoutLen

)

此函数用于解密，将密文输入返回明文

参数：

\_In\_ string private\_key：解密私钥

\_In\_ string n：n值

\_In\_ string c\_data：密文

\_Inout\_ size\_t& inoutLen：密文长度，此值在函数执行完毕后变为加密后的明文长度

返回值：明文字符串

***SHA1***

SHA-1哈希算法处理类

static void Sha1Encode

(

\_In\_ const char\* text,

\_In\_ size\_t len,

\_Inout\_ char\* outData

)

此函数用于对字符进行哈希运算产生哈希值

参数：

\_In\_ const char\* text：要被处理的原数据

\_In\_ size\_t len：原数据长度

\_Inout\_ char\* outData：用于输出经过哈希计算后的十六进制数，以字符串的形式

返回值：无

static bool Sha1StrValidate

(

\_In\_ const char\* input\_1,

\_In\_ const char\* input\_2

)

此函数用于校验两哈希值是否一致

参数：

\_In\_ const char\* input\_1：待被检验的字符串1

\_In\_ const char\* input\_2：待被检验的字符串2

返回值：

相同返回true，不同返回flase