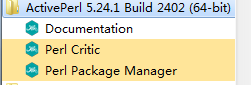
### 这是我的博客的另一个地址：http://blog.sina.com.cn/s/blog\_149e9d2ec0102wzvi.html 2. ActivePerl工具，官网下载链接：<http://www.activestate.com/activeperl/downloads>

1. 首先从网上下载两个文件：一个是Perl，一个是openssl-1.0.1.

[VS2010使用openssl的静态库libeay32.lib和ssleay32.lib](http://photo.blog.sina.com.cn/showpic.html#blogid=149e9d2ec0102wzvi&url=http://album.sina.com.cn/pic/0062ApCkzy7aZtEiTDsfa)  
[VS2010使用openssl的静态库libeay32.lib和ssleay32.lib](http://photo.blog.sina.com.cn/showpic.html#blogid=149e9d2ec0102wzvi&url=http://album.sina.com.cn/pic/0062ApCkzy7aZtEmdC7df)

2. 然后对ActivePerl进行安装。解压openssl-1.0.1.tar.gz至C盘。

[](http://photo.blog.sina.com.cn/showpic.html#blogid=149e9d2ec0102wzvi&url=http://album.sina.com.cn/pic/0062ApCkzy7aZtP6Epd5b)

[VS2010使用openssl的静态库libeay32.lib和ssleay32.lib](http://photo.blog.sina.com.cn/showpic.html#blogid=149e9d2ec0102wzvi&url=http://album.sina.com.cn/pic/0062ApCkzy7aZtNoSBcca)

从http://sourceforge.net/projects/nasm/下载nasm 2.07，将其安装到D:\ProgramFiles\NASM中，并将D:\ProgramFiles\NASM添加到系统环境变量Path中；

6. 在openssl-1.0.1对应的目录下找到这几个文件，他们都是生成对应平台，对应位数的命令操作的帮助文档。上面有一定的帮助作用。不过下面我会进行详细的介绍，或许和这个使用有些不同。注意install.w64 或install.w32

#### [VS2010使用openssl的静态库libeay32.lib和ssleay32.lib](http://photo.blog.sina.com.cn/showpic.html#blogid=149e9d2ec0102wzvi&url=http://album.sina.com.cn/pic/0062ApCkzy7aZuyG4Nk52) 2. 解压openssl。

#### 3. 点击 windows 开始->所有应用->vs2017->开发者命令提示（Developer Command Prompt for 2017）。

#### 4. CD到解压后的openssl的根目录。

**5. 执行命令：**perl Configure VC-WIN64A no-asm--prefix=c:\openssl\_64libRelease

不使用汇编，的release版本

#### 6. ms\do\_ms.bat（创建makefile文件，64位需要使用ms\do\_win64aA.bat）

#### 7. 执行nmake - f ms\ntdll.mak(编译动态库)。

#### 8. 执行nmake - f ms\ntdll.mak test查看库是否有问题。

**9. 执行nmake - f ms\ntdll.mak install 。**

7. 首先我们打开文档INSTALL.W64作为参考。

a:  我们要生成windows下64位release模式的静态库。

首先需要在命令提示窗口依次输入如下命令：

perl Configure VC-WIN64A no-asm--prefix=c:\openssl\_64libRelease

ms\do\_win64a

nmake -f ms\nt.mak

nmake -f ms\nt.makinstall

nmake -f ms\nt.maktest

其中：

VC-WIN64A：表示生成windows下64位release模式的库。

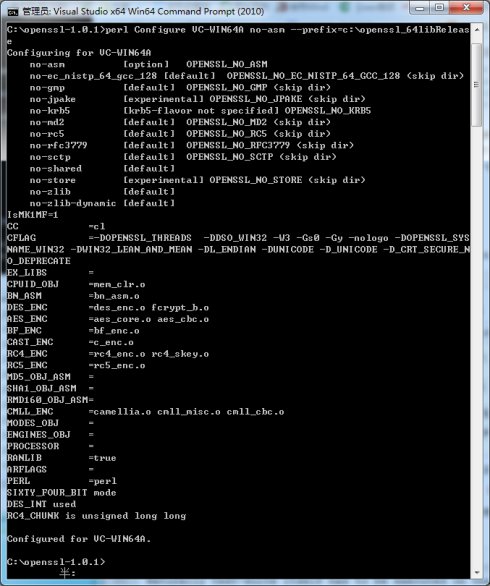
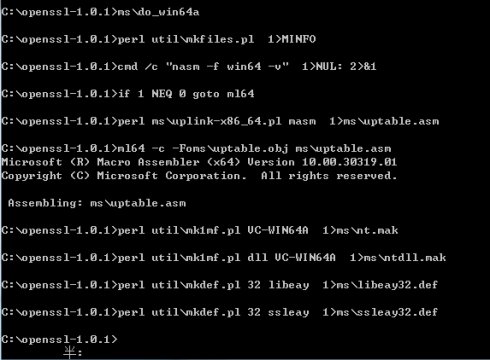
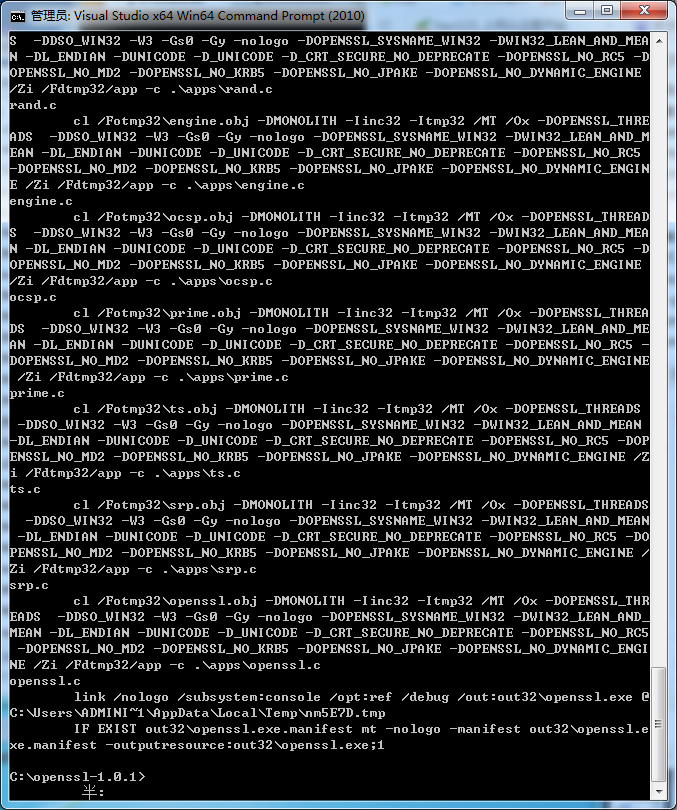
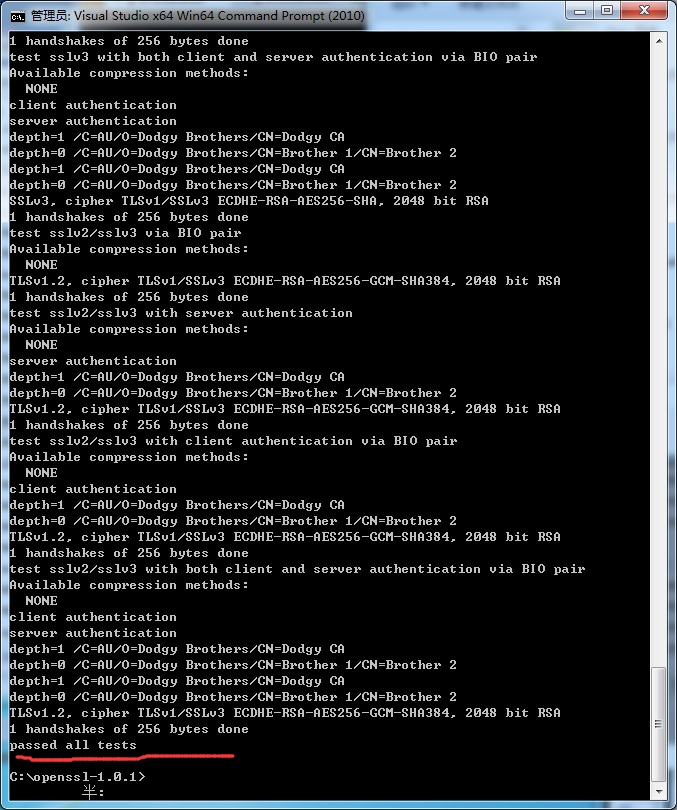
no-asm：表示不使用汇编。

--prefix=c:\openssl\_64Release：表示配置的生成文件都放在目录c:\openssl\_64Release中。

nmake -fms\nt.mak：中nt.mak表示生成静态库，如果换成ntdll.mak表示生成动态库。

nmake -f ms\nt.makinstall：生成的静态库放到c:\openssl\_64libRelease目录下。

下面是这几条命令执行的结果。

[](http://photo.blog.sina.com.cn/showpic.html#blogid=149e9d2ec0102wzvi&url=http://album.sina.com.cn/pic/0062ApCkzy7aZvXMnE09c)  
  
[](http://photo.blog.sina.com.cn/showpic.html#blogid=149e9d2ec0102wzvi&url=http://album.sina.com.cn/pic/0062ApCkzy7aZvXQbw35f)  
  
[](http://photo.blog.sina.com.cn/showpic.html#blogid=149e9d2ec0102wzvi&url=http://album.sina.com.cn/pic/0062ApCkzy7aZvXV2QY24)  
[](http://photo.blog.sina.com.cn/showpic.html#blogid=149e9d2ec0102wzvi&url=http://album.sina.com.cn/pic/0062ApCkzy7aZw42opE06)

b: 如果我们要生成windows下64位debug模式的静态库。

perl Configure debug-VC-WIN64Ano-asm --prefix=c:\openssl\_64libdebug

ms\do\_win64a

nmake -f ms\nt.mak

nmake -f ms\nt.mak install

nmake -f ms\nt.mak test

c: 如果我们要生成windows下64位release模式的动态库。

perl Configure VC-WIN64A no-asm--prefix=c:\openssl\_64dllrelease

ms\do\_win64a

nmake -f ms\ntdll.mak

nmake -f ms\ntdll.mak install

nmake -f ms\ntdll.mak test

d: 如果我们要生成windows下64位debug模式的动态库。

perl Configure debug-VC-WIN64Ano-asm --prefix=c:\openssl\_64dlldebug

ms\do\_win64a

nmake -f ms\ntdll.mak

nmake -f ms\ntdll.mak install

nmake -f ms\ntdll.mak test

**注：**

windows下32位对应的库文件。

a. 我们要生成windows下32位release模式的静态库。

perl Configure VC-WIN32 no-asm --prefix=c:\openssl\_32libRelease

ms\do\_ms

nmake -f ms\nt.mak

nmake -f ms\nt.mak install

nmake -f ms\nt.mak test

b: 如果我们要生成windows下32位debug模式的静态库。

perl Configuredebug-VC-WIN32 no-asm--prefix=c:\openssl\_32libdebug

ms\do\_ms

nmake -f ms\nt.mak

nmake -f ms\nt.mak install

nmake -f ms\nt.mak test

c: 如果我们要生成windows下32位release模式的动态库。

perl Configure VC-WIN32 no-asm--prefix=c:\openssl\_32dllrelease

ms\do\_ms

nmake -f ms\ntdll.mak

nmake -f ms\ntdll.mak install

nmake -f ms\ntdll.mak test

d: 如果我们要生成windows下32位debug模式的动态库。

perl Configuredebug-VC-WIN32 no-asm--prefix=c:\openssl\_32dlldebug

ms\do\_ms

nmake -f ms\ntdll.mak

nmake -f ms\ntdll.mak install

nmake -f ms\ntdll.mak test

之前遇到一个很大的问题，就是如何在VS2010使用openssl编译产生的静态库libeay32.lib和ssleay32.lib。

比如我们写一下代码：

#include “openssl/ssl.h”

#include  “openssl/err.h”

#include “iostream”

using namespace std;

//#pragma comment( lib, "ws2\_32.lib" )  //socket库，关键的一个库

#pragma comment( lib, "libeay32.lib" )

#pragma comment( lib, "ssleay32.lib" )

int main()

{

SSL\_load\_error\_strings();

SSL\_library\_init();

OpenSSL\_add\_all\_algorithms();

ERR\_load\_BIO\_strings();

const SSL\_METHOD\* m = SSLv23\_method();

SSL\_CTX \*ctx = SSL\_CTX\_new(m);

return 0;

}

编译后会出现一大堆问题和错误：

1>------ Rebuild All started: Project: testssl2,Configuration: Debug Win32 ------

1>Build started 2017/5/9 13:44:35.

1>\_PrepareForClean:

1>  Deleting file"Debug\testssl2.lastbuildstate".

1>InitializeBuildStatus:

1>  Creating"Debug\testssl2.unsuccessfulbuild" because "AlwaysCreate" wasspecified.

1>ClCompile:

1>  ssl2.cpp

1>libeay32.lib(bss\_sock.obj) : error LNK2019: unresolvedexternal symbol \_\_imp\_\_closesocket@4 referenced in function\_sock\_free

1>libeay32.lib(bss\_sock.obj) : error LNK2019: unresolvedexternal symbol \_\_imp\_\_shutdown@8 referenced in function\_sock\_free

1>libeay32.lib(bss\_sock.obj) : error LNK2019: unresolvedexternal symbol \_\_imp\_\_recv@16 referenced in function\_sock\_read

1>libeay32.lib(bss\_sock.obj) : error LNK2019: unresolvedexternal symbol \_\_imp\_\_WSASetLastError@4 referenced in function\_sock\_read

1>libeay32.lib(bss\_sock.obj) : error LNK2019: unresolvedexternal symbol \_\_imp\_\_send@16 referenced in function\_sock\_write

1>libeay32.lib(bss\_sock.obj) : error LNK2019: unresolvedexternal symbol \_\_imp\_\_WSAGetLastError@0 referenced in function\_BIO\_sock\_should\_retry

1>E:\VS\TESTSSL1\Debug\testssl2.exe : fatal error LNK1120:6 unresolved externals

1>

1>Build FAILED.

1>

1>Time Elapsed 00:00:01.08

========== Rebuild All: 0 succeeded, 1 failed, 0 skipped==========

我为了这个问题困扰了我好长的时间，现在才现在我缺少了一个关键库的引用，他就是：

**#pragma comment( lib,"ws2\_32.lib" )**

如果我们加上这一句话，那么问题就迎刃而解了。

衷心的希望我曾经的困惑能给你带来帮助。