

Windows환경에서의 OpenSSL설치

(Visual Studio에서 OpenSSL 라이브러리 사용하기)

문서 최초 작성일 : 2007-06-21 < Ver 0.1 >

문서 작성자 : 정은석 이메일 주소 : EunSeok.Jeong@Gmail.com

본 설치 매뉴얼은 Windows 플랫폼 상에서 OpenSSL를 어떻게 설치하고 Visual Studio에서 이 라이브러리를 어떻게 사용하는지 알려주기 위하여 작성하였습니다. 혹시 틀린 내용이 있거나 도움이 될만한 내용이 있으면 언제든지 메일을 보내주시길 바랍니다. ^^
지금부터는 편의상 존칭을 생략하고 적도록 하겠습니다.

[설치 과정]

본 매뉴얼에 따라 OpenSSL을 설치하기 위해서는 다음과 같은 것들을 미리 준비한다.

1. Windows System : Windows 2000 or Windows XP
: 이것은 기본적으로 이미 갖추어진 환경이라 본다.
2. OpenSSL : OpenSSL (Version 0.9.8e)
: OpenSSL은 <http://www.openssl.org/source/> 이곳에서 다운로드 받을 수 있다.
작성시점에서 최신 버전은 openssl-0.9.8e.tar.gz이다.
3. Perl : ActivePerl (Version 5.8.8.820)
: ActivePerl은 <http://www.activestate.com/store/activeperl/download/> 이곳에서 다운로드 받을 수 있다. 작성시점에서 최신 버전은 ActivePerl-5.8.8.820이다. 다운로드를 위해서는 이름과 직장명 정도의 간단한 정보입력을 요구한다.

4. NASM : NASM (Version 0.9.8e)
: Free Netwide Assembler NASM은 아래 경로에서 다운로드 받을 수 있다.
<http://www.kernel.org/pub/software/devel/nasm/binaries/win32/>
작성시점에서 최신 버전은 nasm-0.98.39-win32.zip 이다.

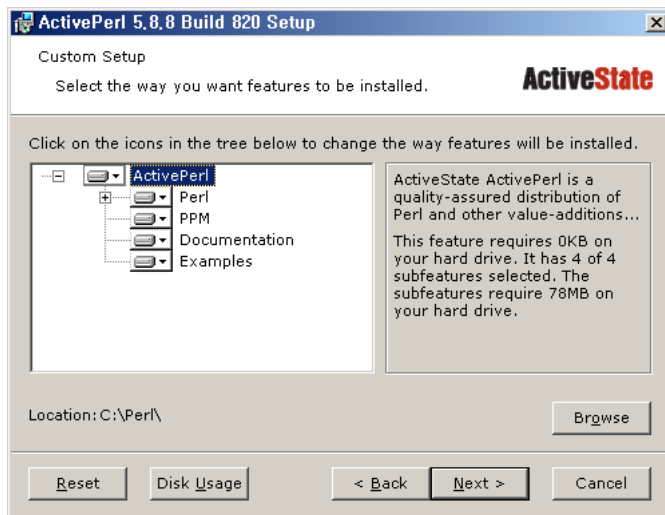
이 매뉴얼의 작성 시점(2007년 6월 21일)에 가장 최신의 버전들을 이용하여 설치하였으나 Major Version Up이 아닌 이상 설치과정에 큰 차이를 보이지는 않을 것으로 생각된다.
위의 것들이 준비되었다면 설치를 시작해보자.

[ActivePerl의 설치 과정]

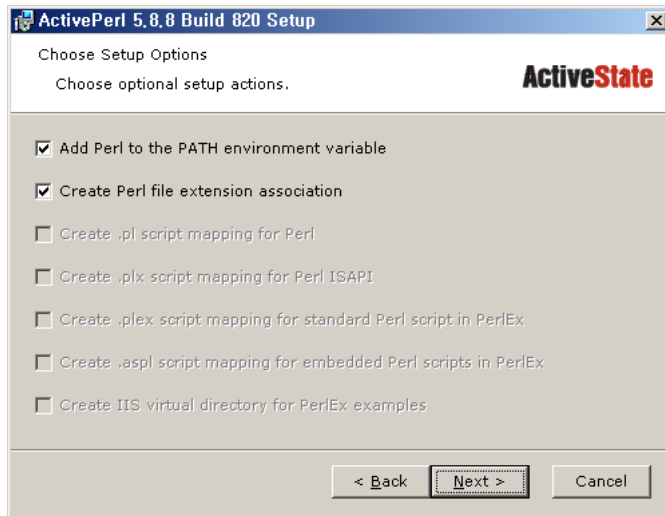
1) 다운로드 받은 설치파일을 실행하면 다음과 같은 화면으로 설치를 시작하게 된다. Perl을 설치하는 이유는 OpenSSL을 설치하기 위해서 Perl Interpreter가 필요하기 때문이다. Next 누름.



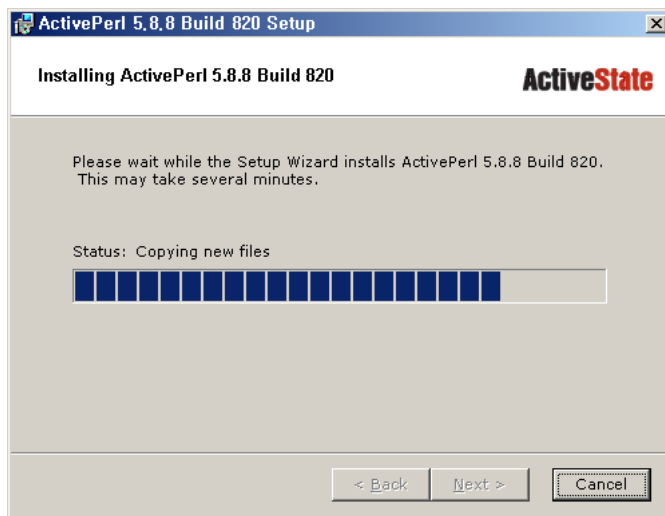
2) 어떠한 구성요소들을 설치할지 선택한다. 기본 경로는 C:\WPerl\로 되어있는데 굳이 꼭 바꾸겠다면 바꾸어도 되지만 기본값으로 그냥 두고 설치하기를 추천한다. Next 누름.



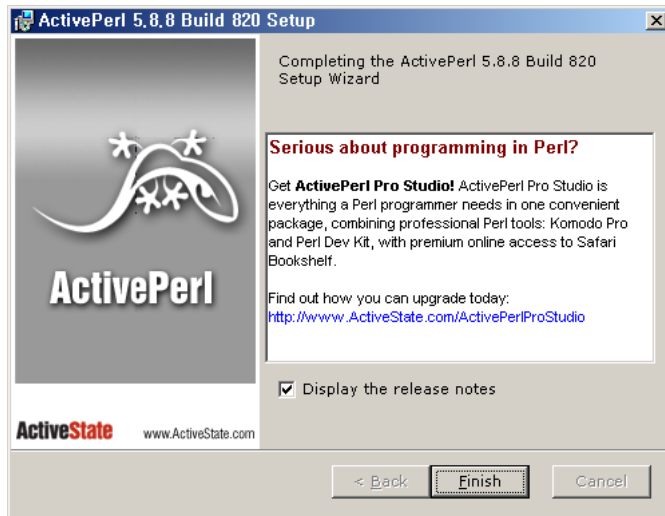
3) Perl을 Path 환경변수에 더할 것인가 그리고 Perl파일의 확장자(즉, *.pl과 같은)를 등록 하겠는가를 선택하는 화면이다. 마찬가지로 둘 다 체크된 기본값으로 두고 Next 누름.



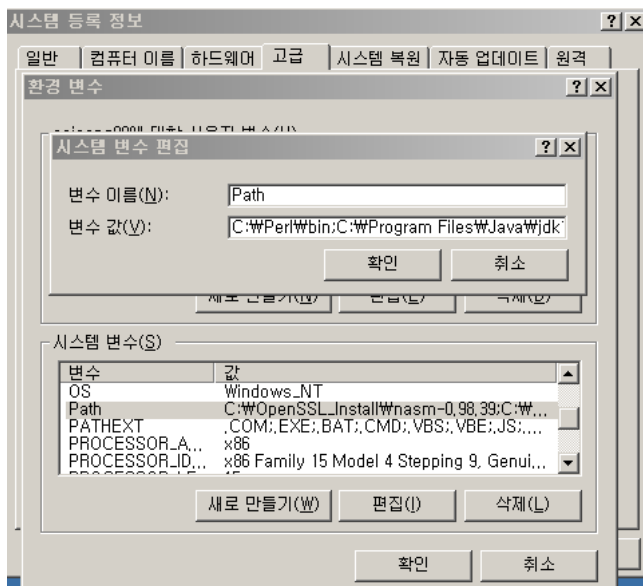
4) Perl의 설치가 진행된다. Generating HTML documentation 단계에서 시간이 좀 오래 소요되니 중지하지 말고 기다리자.



5) 설치 종료. 아래와 같은 화면이 뜨면 Perl의 설치가 완료된 것이다. Release Notes는 꼭 읽어야 하는 것은 아니다. 본인의 선택에 따라 결정하고 Perl의 설치를 마친다.

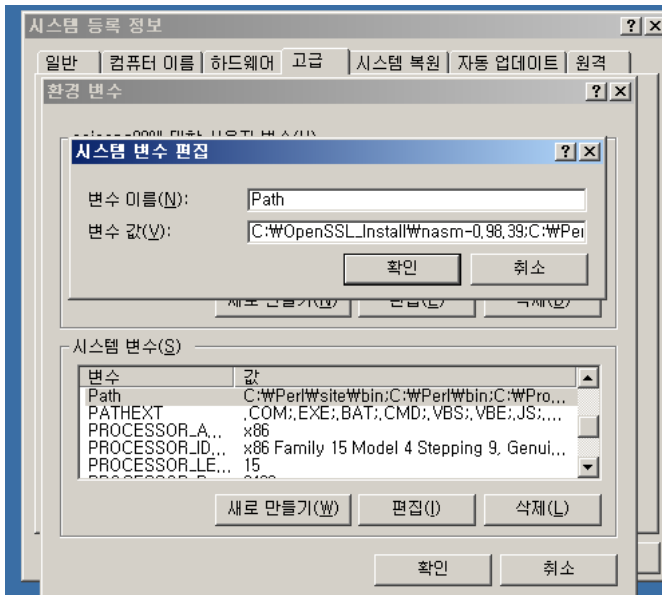


6) Perl이 시스템의 Path 환경변수에 등록이 정상적으로 되었는지 확인한다. 문제없이 설치 되었다면 Path 환경변수의 제일 앞에 C:\Perl\bin;과 같은 내용이 추가되었을 것이다. 혹시 C:\Perl\site\bin;과 같은 내용도 추가되었을 수도 있는데 어차피 이건 Path 경로이기 때문에 그냥 두어도 상관없다. 만약 이러한 내용이 추가되지 않았다면 꼭 추가해 주도록 한다. 그렇지 않으면 나중에 OpenSSL을 설치할 때 문제가 생기게 된다.



[NASM의 설치 과정]

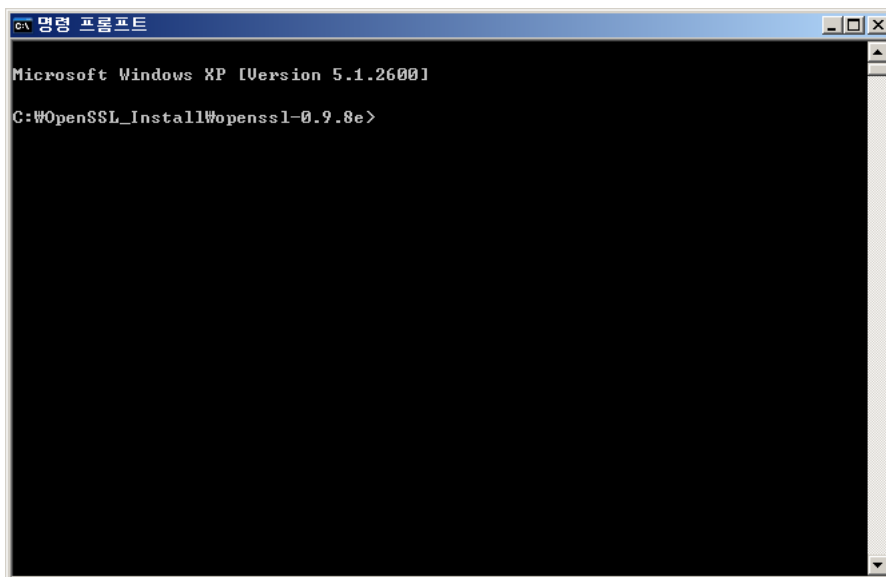
1) 어셈블리 컴파일러가 필요한 이유는 OpenSSL의 설치시에 속도에 큰 영향을 미치는 부분을 어셈블리로 컴파일 하기 위해서이다. 다들 아시다시피 성능에 큰 영향을 미치는 부분이 어셈블리로 작성되었을 경우 얻어지는 성능향상은 무시하지 못할만큼 꽤 크다. OpenSSL 역시 암호화 기능으로 인해 많은 연산을 요구하므로 어셈블리로 그런 부분을 처리하여 속도향상을 꾀한 것 같다. 다운로드 받은 NASM을 적절한 경로에 풀어주고 해당 경로를 시스템의 Path 환경변수에 등록한다. 글쓴이의 경우 OpenSSL 설치에 필요한 것들을 C:\WOpenSSL_Install이란 곳에 모아놓고 설치하였으므로 아래와 같은 경로를 등록하였다.



[OpenSSL의 설치 과정]

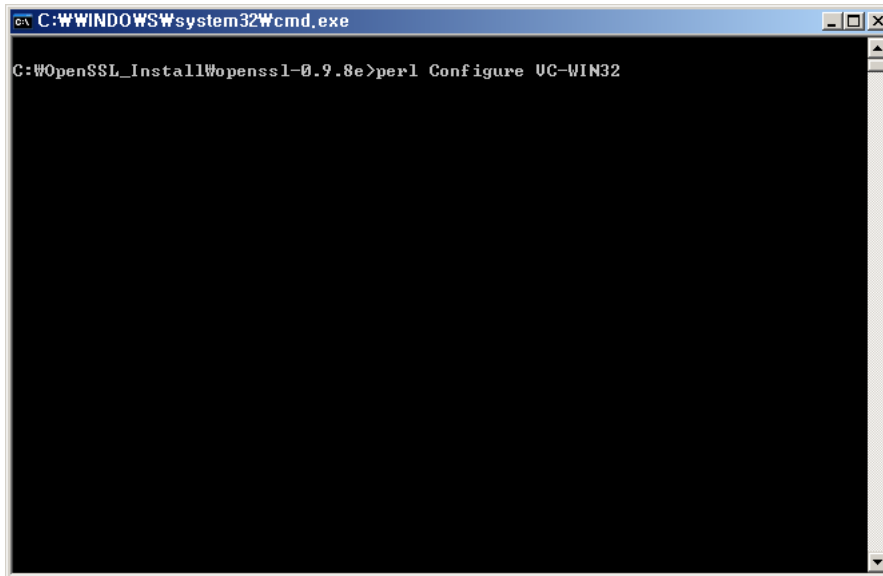
여기까지 문제없이 왔다면 OpenSSL 설치를 위한 기본적인 환경은 구축된 셈이다.

1) 명령어 프롬프트 창을 하나 띄운다.



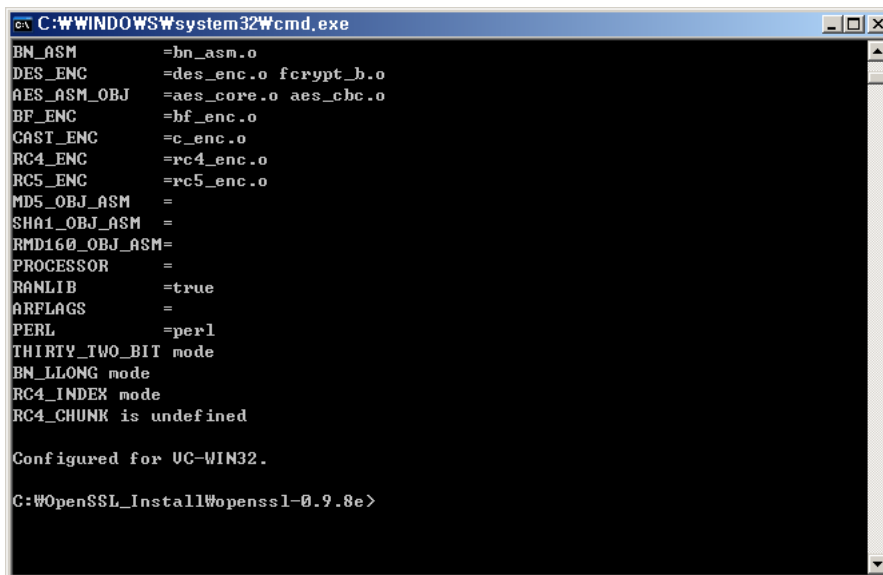
2) (Configure 과정) OpenSSL을 적당한 경로에 압축을 풀어놓고 해당 경로에서 다음과 같은 명령어를 수행한다.

명령어 : perl Configure VS-WIN32



```
C:\WINDOWS\system32\cmd.exe
C:\OpenSSL\Install\openssl-0.9.8e>perl Configure UC-WIN32
```

3) 위의 명령어를 수행하여 정상적으로 처리되면 아래 그림과 같이 나올 것이다.

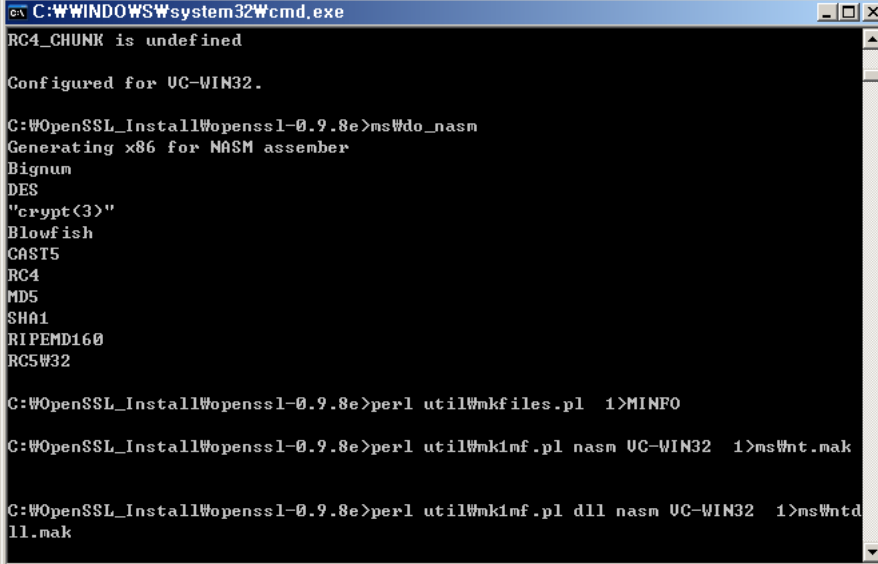


```
C:\WINDOWS\system32\cmd.exe
BN_ASM          =bn_asm.o
DES_ENC         =des_enc.o fcrypt_b.o
AES_ASM_OBJ     =aes_core.o aes_chc.o
BF_ENC         =bf_enc.o
CAST_ENC       =c_enc.o
RC4_ENC        =rc4_enc.o
RC5_ENC        =rc5_enc.o
MD5_OBJ_ASM    =
SHA1_OBJ_ASM   =
RMD160_OBJ_ASM=
PROCESSOR      =
RANLIB         =true
ARFLAGS        =
PERL           =perl
THIRTY_TWO_BIT mode
BN_LLONG mode
RC4_INDEX mode
RC4_CHUNK is undefined

Configured for UC-WIN32.
C:\OpenSSL\Install\openssl-0.9.8e>
```

4) (Makefile 생성 과정) 이제 다음 명령어를 수행한다.

명령어 : msWdo_nasm



```
C:\WINDOWS\system32\cmd.exe
RC4_CHUNK is undefined

Configured for UC-WIN32.

C:\OpenSSL\Install\openssl-0.9.8e>msWdo_nasm
Generating x86 for NASM assembler
Bignum
DES
"crypt(3)"
Blowfish
CAST5
RC4
MD5
SHA1
RIPEMD160
RC5W32

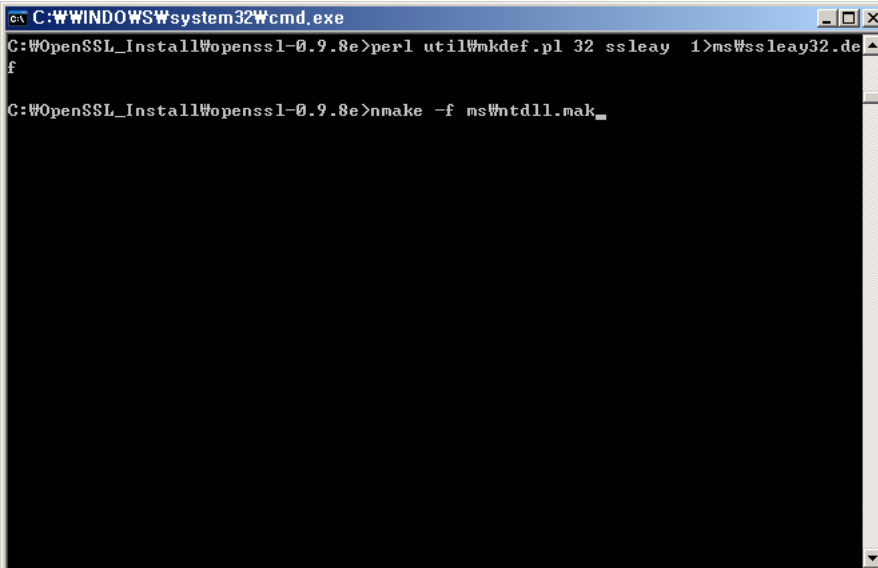
C:\OpenSSL\Install\openssl-0.9.8e>perl util\mkfiles.pl 1>MINFO

C:\OpenSSL\Install\openssl-0.9.8e>perl util\mkimf.pl nasm UC-WIN32 1>msWnt.mak

C:\OpenSSL\Install\openssl-0.9.8e>perl util\mkimf.pl dll nasm UC-WIN32 1>msWntdll.mak
```

5) (Building 과정) 위의 명령어를 수행하여 정상적으로 처리되면 아래 그림과 같이 나올 것이다. 그러면 이제 다음 명령을 수행한다.

명령어 : nmake -f msWntdll.mak



```
C:\WINDOWS\system32\cmd.exe
C:\OpenSSL\Install\openssl-0.9.8e>perl util\mkdef.pl 32 ssleay 1>msWssleay32.def
f
C:\OpenSSL\Install\openssl-0.9.8e>nmake -f msWntdll.mak
```

6) 위의 명령어를 수행하면 아래 그림과 같은 내용이 쭉 나오면서 계속 컴파일 될 것이다.
컴파일 하는데 시간이 제법 소요된다. 완전히 작업이 끝날 때까지 기다리자.

```

C:\WINDOWS\system32\cmd.exe - nmake -f msWntdll.mak
Copying: ./crypto/o_dir.h to tmp32dll/o_dir.h
perl util/copy.pl .\crypto\md4\md4_loc1.h tmp32dll\md4_loc1.h
Copying: ./crypto/md4/md4_loc1.h to tmp32dll/md4_loc1.h
perl util/copy.pl .\crypto\md5\md5_loc1.h tmp32dll\md5_loc1.h
Copying: ./crypto/md5/md5_loc1.h to tmp32dll/md5_loc1.h
perl util/copy.pl .\crypto\sha\sha_loc1.h tmp32dll\sha_loc1.h
Copying: ./crypto/ripemd\ripemd_loc1.h tmp32dll\ripemd_loc1.h
perl util/copy.pl .\crypto\ripemd\ripemdconst.h tmp32dll\ripemdconst.h
Copying: ./crypto/ripemd\ripemdconst.h to tmp32dll\ripemdconst.h
perl util/copy.pl .\crypto\des\des_loc1.h tmp32dll\des_loc1.h
Copying: ./crypto/des/des_loc1.h to tmp32dll/des_loc1.h
perl util/copy.pl .\crypto\des\des_rpc_des.h tmp32dll\des_rpc_des.h
Copying: ./crypto/des/des_rpc_des.h to tmp32dll\des_rpc_des.h
perl util/copy.pl .\crypto\des\des_spr.h tmp32dll\des_spr.h
Copying: ./crypto/des/des_spr.h to tmp32dll\des_spr.h
perl util/copy.pl .\crypto\des\des_ver.h tmp32dll\des_ver.h
Copying: ./crypto/des/des_ver.h to tmp32dll\des_ver.h
perl util/copy.pl .\crypto\rc2\rc2_loc1.h tmp32dll\rc2_loc1.h
Copying: ./crypto/rc2/rc2_loc1.h to tmp32dll\rc2_loc1.h
perl util/copy.pl .\crypto\rc4\rc4_loc1.h tmp32dll\rc4_loc1.h
Copying: ./crypto/rc4/rc4_loc1.h to tmp32dll\rc4_loc1.h
perl util/copy.pl .\crypto\idea\idea_loc1.h tmp32dll\idea_loc1.h

```

7) 컴파일을 모두 마치고 나면 해당 경로에서 다음과 같은 명령어를 수행하여 아래 파일들이 생성되었는지 확인한다. 생성되는 파일들은 모든 out32dll안에 생성된다.

dir out32dll*.dll out32dll*.lib

생성되어야 하는 파일들 : libeay32.dll, ssleay32.dll, libeay32.lib, ssleay32.lib

```

C:\WINDOWS\system32\cmd.exe
C:\OpenSSL\Install\openssl-0.9.8e>
C:\OpenSSL\Install\openssl-0.9.8e>
C:\OpenSSL\Install\openssl-0.9.8e>
C:\OpenSSL\Install\openssl-0.9.8e>dir out32dll\*.dll out32dll\*.lib
C 드라이브의 볼륨: SYSTEM
볼륨 일련 번호: 8CBF-B6F8

C:\OpenSSL\Install\openssl-0.9.8e\out32dll 디렉터리

2007-06-08 오전 06:02          1,077,248 libeay32.dll
2007-06-08 오전 06:03          200,704 ssleay32.dll

C:\OpenSSL\Install\openssl-0.9.8e\out32dll 디렉터리

2007-06-08 오전 06:02          664,738 libeay32.lib
2007-06-08 오전 06:03           52,202 ssleay32.lib
4개 파일              1,994,892 바이트
0개 디렉터리          15,083,421,696 바이트 남음

C:\OpenSSL\Install\openssl-0.9.8e>

```


8) (Test 과정) 4개의 파일들이 생겼다면 이제 테스트를 해본다. 압축 폰 디렉토리안의 out32dll이라는 폴더안으로 들어간다. 그리고 그 안에서 다음 명령어를 실행하여 테스트를 수행한다.

명령어 : ..WmsWtest

여기서 기억해야 할 것은 꼭 out32dll 디렉토리 안에 들어가서 테스트 명령을 실행해야 한다는 점이다. 그렇지 않고 그냥 압축 폰 디렉토리 그 상태에서 명령어를 수행하면 아래 두 번째 있는 그림처럼 에러가 나며 테스트가 진행되지 않는다.

```
C:\WINDOWS\system32\cmd.exe
C:\OpenSSL\Install\openssl-0.9.8e>
C:\OpenSSL\Install\openssl-0.9.8e>
C:\OpenSSL\Install\openssl-0.9.8e>cd out32dll

C:\OpenSSL\Install\openssl-0.9.8e\out32dll>..WmsWtest
rsa_test
PKCS #1 v1.5 encryption/decryption ok
OAEP encryption/decryption ok
PKCS #1 v1.5 encryption/decryption ok
OAEP encryption/decryption ok
PKCS #1 v1.5 encryption/decryption ok
OAEP encryption/decryption ok
PKCS #1 v1.5 encryption/decryption ok
OAEP encryption/decryption ok
PKCS #1 v1.5 encryption/decryption ok
OAEP encryption/decryption ok
PKCS #1 v1.5 encryption/decryption ok
OAEP encryption/decryption ok
destest
Doing cbcm
Doing ech
Doing ede ech
Doing cbc
Doing desx cbc
Doing ede cbc
```

```
C:\WINDOWS\system32\cmd.exe
C:\OpenSSL\Install\openssl-0.9.8e>
C:\OpenSSL\Install\openssl-0.9.8e>
C:\OpenSSL\Install\openssl-0.9.8e>
C:\OpenSSL\Install\openssl-0.9.8e>
C:\OpenSSL\Install\openssl-0.9.8e>
C:\OpenSSL\Install\openssl-0.9.8e>
C:\OpenSSL\Install\openssl-0.9.8e>nsWtest
rsa_test
'rsa_test'은<는> 내부 또는 외부 명령, 실행할 수 있는 프로그램, 또는
배치 파일이 아닙니다.
problems.....

C:\OpenSSL\Install\openssl-0.9.8e>
C:\OpenSSL\Install\openssl-0.9.8e>
C:\OpenSSL\Install\openssl-0.9.8e>
```

10) 모든 것이 정상적으로 설치되었다면 테스트 과정은 성공적으로 마치고 passed all tests라는 문구가 나오게 된다. 이 문구를 보았다면 OpenSSL의 설치를 성공적으로 수행한 것이다. 축하한다! ^^;

```

C:\WINDOWS\system32\cmd.exe
NONE
client authentication
depth=1 /C=AU/O=Dodgy Brothers/CN=Dodgy CA
depth=0 /C=AU/O=Dodgy Brothers/CN=Brother 1/CN=Brother 2
TLSv1, cipher TLSv1/SSLv3 DHE-RSA-AES256-SHA, 512 bit RSA
1 handshakes of 256 bytes done
test sslv2/sslv3 with both client and server authentication via BIO pair
Available compression methods:
NONE
server authentication
depth=1 /C=AU/O=Dodgy Brothers/CN=Dodgy CA
depth=0 /C=AU/O=Dodgy Brothers/CN=Brother 1/CN=Brother 2
depth=1 /C=AU/O=Dodgy Brothers/CN=Dodgy CA
depth=0 /C=AU/O=Dodgy Brothers/CN=Brother 1/CN=Brother 2
TLSv1, cipher TLSv1/SSLv3 DHE-RSA-AES256-SHA, 512 bit RSA
1 handshakes of 256 bytes done
passed all tests

C:\OpenSSL\Install\openssl-0.9.8e\out32dll>_

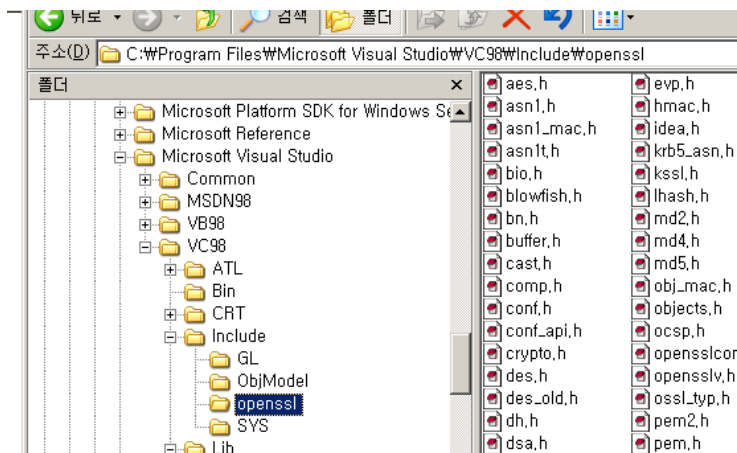
```

[Visual Studio에서 OpenSSL 사용하기]

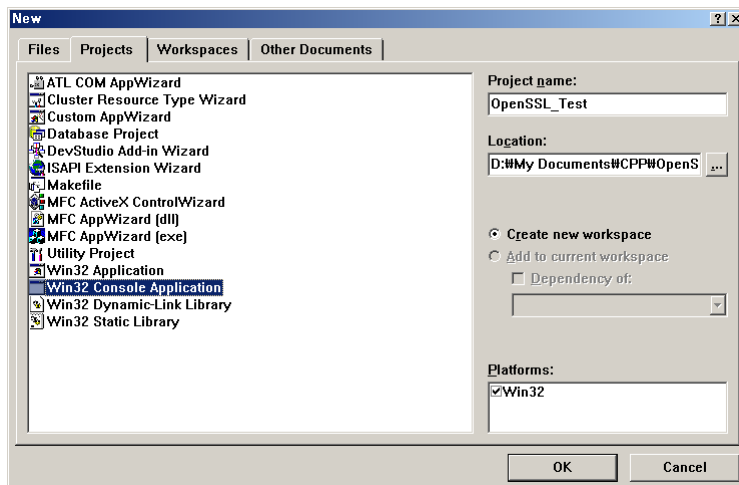
1) 이제 OpenSSL은 다 설치되었기 때문에 Visual Studio에서 사용하기만 하면 된다.

우선 OpenSSL을 설치하던 곳(즉, OpenSSL을 압축풀어 놓았던 곳)의 헤더파일들을 Visual Studio의 include 디렉토리 안에 복사해 넣는다.

openssl-0.9.8e\include\ 이 안에 보면 openssl이라는 디렉토리가 있고 그 안에 보면 수십개의 헤더 파일들이 존재한다. 이 openssl 디렉토리를 통째로 복사하여 C:\Program Files\Microsoft Visual Studio\VC98\Include 이 안에 넣어준다. 이것을 수행하고 난 뒤의 디렉토리 구성은 다음과 같다.



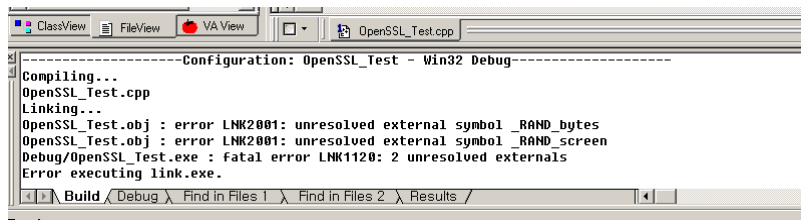
2) Visual Studio를 실행하여 Win32 Console Application으로 새 프로젝트를 하나 만들어 empty project를 연다.



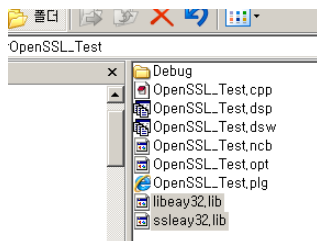
3) 다음과 같은 소스 코드를 입력한다.

```
////////////////////////////////////  
#include <stdio.h>  
#include <openssl/ssl.h>  
#include <openssl/rand.h>  
  
void main()  
{  
    int i = 0;  
    int iSize = 64;  
    unsigned char* uBuff = NULL;  
  
    uBuff = (unsigned char*)malloc(sizeof(unsigned char) * iSize);  
  
    RAND_screen();  
  
    RAND_bytes(uBuff, iSize);  
  
    printf("Rand() is...\\n");  
    for (i = 0; i < iSize; i++)  
    {  
        printf("%d\\n", (int)uBuff[i]);  
    }  
  
    delete uBuff;  
}  
////////////////////////////////////
```

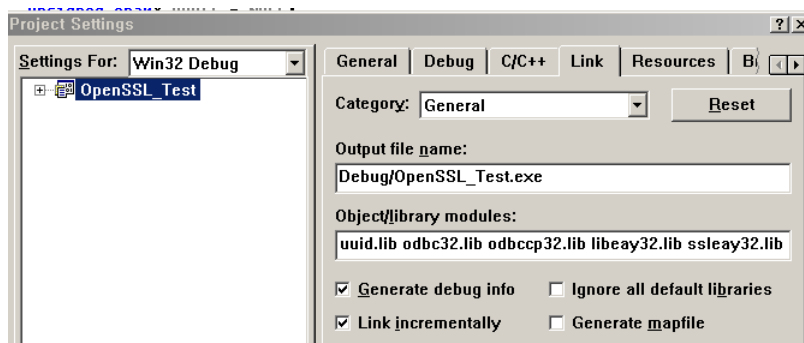
4) Ctrl+F5를 눌러보면 컴파일은 되는데 몇가지 Link관련 에러가 나는 것을 확인할 수 있다. 이것은 OpenSSL 라이브러리를 연결하지 않아서 그런 것이다.



5) 아까 OpenSSL 소스파일이 있던 경로에 가서 out32dll안에 보면 libeay32.lib, ssleay32.lib, libeay32.dll, ssleay32.dll 이 파일들이 있는데 확장자가 lib인것은 정적 라이브러리, dll인것은 동적 라이브러리이다. 테스트를 위하여 libeay32.lib, ssleay32.lib 이 두 파일을 복사하여 OpenSSL_Test라고 만들었던 Visual Studio 프로젝트 디렉토리 안에 붙여넣는다.



그리고 Visual Studio 메뉴에서 Project -> Settings -> Link -> Object/library modules 란에 libeay32.lib ssleay32.lib를 추가로 입력해 준다.



6) 확인을 눌러 Project Settings 창을 닫고 다시 Ctrl+F5를 눌러보자. 에러 없이 잘 수행되는 것을 확인할 수 있다. 만약 다시 에러가 난다면 프로젝트 디렉토리에 Debug 디렉토리를 통째로 지운후 다시 시도해본다. 테스트한 소스의 내용은 OpenSSL 라이브러리를 이용하여 Random Number를 출력한 것이다. 아래 그림처럼 랜덤한 숫자들이 출력된다면 성공한 것이다.

```
C:\D:\My Documents\WCPP\WOpenSSL_Test\WDebug\WOpenSSL_Test.exe
Rand() is...
42
168
148
131
178
90
228
76
170
18
29
56
211
109
57
234
151
83
145
51
160
194
79
39
```

[맺음말]

여기까지 모든 과정이 문제없이 진행되었다면 OpenSSL을 성공적으로 설치하고 Visual Studio에서 이를 사용하는 가장 기본적인 방법까지 익힌 것입니다. 축하합니다~! ^^; 이제 앞으로 남은 일은 이것들을 열심히 다루어 보고 OpenSSL라이브러리를 완전히 자기것으로 익히는 것이겠죠? 앞으로의 험난한 여정을 위해 설치 매뉴얼은 이것으로 끝을 내도록 하겠습니다. 그럼 이만~ ^^ /

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

문서 최초 작성일 : 2007-06-21 < Ver 0.1 >

문서 작성자 : 정은석 이메일 주소 : EunSeok.Jeong@Gmail.com