# windows

## 命令

set PATH=%PATH%;C:\ProgramFiles\NASM;

perl Configure VC-WIN32

ms\do\_nasm

nmake –f ms\ntdll.mak

nmake -f ms\ntdll.mak install

## 输出

inc32

out32dll

Compilation and Installation

The following page is a combination of the INSTALL file provided with the OpenSSL library and notes from the field. If you have questions about what you are doing or seeing, then you should consultINSTALL since it contains the commands and specifies the behavior by the development team.

OpenSSL uses a custom build system to configure the library. Configuration will allow the library to set up the recursive makefiles from makefile.org. Once configured, you use make to build the library.

There are two generations of build system. First is the build system used in OpenSSL 1.0.2 and below. The instructions below apply to it. Second is the build system for OpenSSL 1.1.0 and above. Its currently available in Master through Git. The instructions are similar, but not the same. For example, the second generation abandons the monolithic Configure and places individual configurations in the Configurations directory.

After you configure and build the library, you should always perform a make test to ensure the library performs as expected under its self tests. If you are building OpenSSL 1.1.0 and above, then you will also need PERL 5.10 or high (see README.PERL for details).

OpenSSL's build system does not rely upon autotools or libtool. Also see [Why aren't tools like 'autoconf' and 'libtool' used?](http://www.openssl.org/support/faq.html#MISC5) in the OpenSSL FAQ.

|  |
| --- |
| **Contents**   [[hide](https://wiki.openssl.org/index.php/Compilation_and_Installation)]   * [1 Retrieve source code](https://wiki.openssl.org/index.php/Compilation_and_Installation#Retrieve_source_code) * [2 Configuration](https://wiki.openssl.org/index.php/Compilation_and_Installation#Configuration)   + [2.1 Supported Platforms](https://wiki.openssl.org/index.php/Compilation_and_Installation#Supported_Platforms)   + [2.2 Configure & Config](https://wiki.openssl.org/index.php/Compilation_and_Installation#Configure_.26_Config)   + [2.3 Dependencies](https://wiki.openssl.org/index.php/Compilation_and_Installation#Dependencies) * [3 Configure Options](https://wiki.openssl.org/index.php/Compilation_and_Installation#Configure_Options)   + [3.1 PREFIX and OPENSSLDIR](https://wiki.openssl.org/index.php/Compilation_and_Installation#PREFIX_and_OPENSSLDIR)   + [3.2 Debug Configuration](https://wiki.openssl.org/index.php/Compilation_and_Installation#Debug_Configuration)   + [3.3 Modifying Build Settings](https://wiki.openssl.org/index.php/Compilation_and_Installation#Modifying_Build_Settings)   + [3.4 Using RPATHs](https://wiki.openssl.org/index.php/Compilation_and_Installation#Using_RPATHs)   + [3.5 FIPS Capable Library](https://wiki.openssl.org/index.php/Compilation_and_Installation#FIPS_Capable_Library)   + [3.6 Compile Time Checking](https://wiki.openssl.org/index.php/Compilation_and_Installation#Compile_Time_Checking) * [4 Compilation](https://wiki.openssl.org/index.php/Compilation_and_Installation#Compilation)   + [4.1 Quick](https://wiki.openssl.org/index.php/Compilation_and_Installation#Quick) * [5 Platfom specific](https://wiki.openssl.org/index.php/Compilation_and_Installation#Platfom_specific)   + [5.1 Linux](https://wiki.openssl.org/index.php/Compilation_and_Installation#Linux)     - [5.1.1 Intel](https://wiki.openssl.org/index.php/Compilation_and_Installation#Intel)     - [5.1.2 ARM](https://wiki.openssl.org/index.php/Compilation_and_Installation#ARM)     - [5.1.3 X32 (ILP32)](https://wiki.openssl.org/index.php/Compilation_and_Installation#X32_.28ILP32.29)   + [5.2 Windows](https://wiki.openssl.org/index.php/Compilation_and_Installation#Windows)     - [5.2.1 W32 / Windows NT - Windows 9x](https://wiki.openssl.org/index.php/Compilation_and_Installation#W32_.2F_Windows_NT_-_Windows_9x)     - [5.2.2 W64](https://wiki.openssl.org/index.php/Compilation_and_Installation#W64)     - [5.2.3 Windows CE](https://wiki.openssl.org/index.php/Compilation_and_Installation#Windows_CE)   + [5.3 OS X](https://wiki.openssl.org/index.php/Compilation_and_Installation#OS_X)   + [5.4 iOS](https://wiki.openssl.org/index.php/Compilation_and_Installation#iOS)   + [5.5 Android](https://wiki.openssl.org/index.php/Compilation_and_Installation#Android)   + [5.6 More](https://wiki.openssl.org/index.php/Compilation_and_Installation#More)     - [5.6.1 VAX/VMS](https://wiki.openssl.org/index.php/Compilation_and_Installation#VAX.2FVMS)     - [5.6.2 OS/2](https://wiki.openssl.org/index.php/Compilation_and_Installation#OS.2F2)     - [5.6.3 NetWare](https://wiki.openssl.org/index.php/Compilation_and_Installation#NetWare)     - [5.6.4 HP-UX](https://wiki.openssl.org/index.php/Compilation_and_Installation#HP-UX) * [6 Autoconf](https://wiki.openssl.org/index.php/Compilation_and_Installation#Autoconf) |

[[edit](https://wiki.openssl.org/index.php?title=Compilation_and_Installation&action=edit&section=1)]Retrieve source code

The OpenSSL source code can be downloaded from [OpenSSL Source Tarballs](http://www.openssl.org/source/) or any suitable [ftp mirror](http://www.openssl.org/source/mirror.html). There are various versions including stable as well as unstable versions.

The source code is managed via Git. Its referred to as Master. The repository is

<git://git.openssl.org/openssl.git>

The source is also available via a [GitHub](https://github.com/openssl/openssl) mirror. This repository is updated every 15 minutes.

* [Accessing OpenSSL source code via Git](https://wiki.openssl.org/index.php/Use_of_Git)

[[edit](https://wiki.openssl.org/index.php?title=Compilation_and_Installation&action=edit&section=2)]Configuration

OpenSSL is configured for a particular platform with protocol and behavior options using Configure and config.

[[edit](https://wiki.openssl.org/index.php?title=Compilation_and_Installation&action=edit&section=3)]**Supported Platforms**

You can run Configure LIST to see a list of available platforms.

$ ./Configure LIST

BC-32

BS2000-OSD

BSD-generic32

BSD-generic64

BSD-ia64

BSD-sparc64

BSD-sparcv8

BSD-x86

BSD-x86-elf

BSD-x86\_64

Cygwin

Cygwin-x86\_64

DJGPP

...

If your platform is not listed, then use a similar platform and tune the $cflags and $ldflags by making a copy of the configure line and giving it its own name. $cflags and $ldflags correspond to fields 2 and 6 in a configure line. An example of using a similar configure line is presented in [Using RPATHs](https://wiki.openssl.org/index.php/Compilation_and_Installation#Using_RPATHs).

[[edit](https://wiki.openssl.org/index.php?title=Compilation_and_Installation&action=edit&section=4)]**Configure & Config**

You use Configure and config to tune the compile and installation process through options and switches. The difference between is Configure properly handles the host-arch-compiler triplet, and configdoes not. config attempts to guess the triplet, so its a lot like autotool's config.guess.

You can usually use config and it will do the right thing (from Ubuntu 13.04, x64):

$ ./config

Operating system: x86\_64-whatever-linux2

Configuring for linux-x86\_64

Configuring for linux-x86\_64

no-ec\_nistp\_64\_gcc\_128 [default] OPENSSL\_NO\_EC\_NISTP\_64\_GCC\_128 (skip dir)

no-gmp [default] OPENSSL\_NO\_GMP (skip dir)

no-jpake [experimental] OPENSSL\_NO\_JPAKE (skip dir)

no-krb5 [krb5-flavor not specified] OPENSSL\_NO\_KRB5

...

Mac OS X can have issues (its often a neglected platform), and you will have to use Configure:

./Configure darwin64-x86\_64-cc

Configuring for darwin64-x86\_64-cc

no-ec\_nistp\_64\_gcc\_128 [default] OPENSSL\_NO\_EC\_NISTP\_64\_GCC\_128 (skip dir)

no-gmp [default] OPENSSL\_NO\_GMP (skip dir)

no-jpake [experimental] OPENSSL\_NO\_JPAKE (skip dir)

no-krb5 [krb5-flavor not specified] OPENSSL\_NO\_KRB5

...

You can also configure on Darwin by exporting KERNEL\_BITS:

$ export KERNEL\_BITS=64

$ ./config shared no-ssl2 no-ssl3 enable-ec\_nistp\_64\_gcc\_128 --openssldir=/usr/local/ssl/macosx-x64/

Operating system: i686-apple-darwinDarwin Kernel Version 12.5.0: Sun Sep 29 13:33:47 PDT 2013; root:xnu-2050.48.12~1/RELEASE\_X86\_64

Configuring for darwin64-x86\_64-cc

Configuring for darwin64-x86\_64-cc

no-gmp [default] OPENSSL\_NO\_GMP (skip dir)

no-jpake [experimental] OPENSSL\_NO\_JPAKE (skip dir)

no-krb5 [krb5-flavor not specified] OPENSSL\_NO\_KRB5

...

If you provide a option not known to configure or ask for help, then you get a brief help message:

$ ./Configure --help

Usage: Configure [no-<cipher> ...] [enable-<cipher> ...] [experimental-<cipher> ...]

[-Dxxx] [-lxxx] [-Lxxx] [-fxxx] [-Kxxx] [no-hw-xxx|no-hw] [[no-]threads] [[no-]shared]

[[no-]zlib|zlib-dynamic] [no-asm] [no-dso] [no-krb5] [sctp] [386] [--prefix=DIR]

[--openssldir=OPENSSLDIR] [--with-xxx[=vvv]] [--test-sanity] os/compiler[:flags]

And if you supply an unknown triplet:

$ ./Configure darwin64-x86\_64-clang

Configuring for darwin64-x86\_64-clang

Usage: Configure [no-<cipher> ...] [enable-<cipher> ...] [experimental-<cipher> ...]

[-Dxxx] [-lxxx] [-Lxxx] [-fxxx] [-Kxxx] [no-hw-xxx|no-hw] [[no-]threads] [[no-]shared]

[[no-]zlib|zlib-dynamic] [no-asm] [no-dso] [no-krb5] [sctp] [386] [--prefix=DIR]

[--openssldir=OPENSSLDIR] [--with-xxx[=vvv]] [--test-sanity] os/compiler[:flags]

pick os/compiler from:

BC-32 BS2000-OSD BSD-generic32 BSD-generic64 BSD-ia64 BSD-sparc64 BSD-sparcv8

BSD-x86 BSD-x86-elf BSD-x86\_64 Cygwin Cygwin-pre1.3 DJGPP MPE/iX-gcc OS2-EMX

...

NOTE: If in doubt, on Unix-ish systems use './config'.

[[edit](https://wiki.openssl.org/index.php?title=Compilation_and_Installation&action=edit&section=5)]**Dependencies**

If you are prompted to run make depend, then you must do so. For OpenSSL 1.0.2 and below, its required to update the standard distribution once configuration options change.

Since you've disabled or enabled at least one algorithm, you need to do

the following before building:

make depend

OpenSSL 1.1.0 and above performs the dependency step for you, so you should not see the message. However, you should perform a make clean to ensure the list of objects files is accurate after a reconfiguration.

[[edit](https://wiki.openssl.org/index.php?title=Compilation_and_Installation&action=edit&section=6)]Configure Options

OpenSSL has been around a long time, and it carries around a lot of cruft. For example, from above, SSLv2 is enabled by default. SSLv2 is completely broken, and you should disable it during configuration. You can disable protocols and provide other options through Configure and config, and the following lists some of them.

**Note**: if you specify a non-existent option, then the configure scripts will proceed without warning. For example, if you inadvertently specify **no-sslv2** rather than **no-ssl2 no-ssl3**, the script will configure *with* SSLv2 and *without* warning for the unknown no-sslv2.

| **OpenSSL Library Options** | |
| --- | --- |
| **Option** | **Description** |
| --prefix=XXX | See [PREFIX and OPENSSLDIR](https://wiki.openssl.org/index.php/Compilation_and_Installation#PREFIX_and_OPENSSLDIR) in the next section (below). |
| --openssldir=XXX | See [PREFIX and OPENSSLDIR](https://wiki.openssl.org/index.php/Compilation_and_Installation#PREFIX_and_OPENSSLDIR) in the next section (below). |
| -d | Debug build of the library. Optimizations are disabled (no -O3 or similar) and libefence is used (apt-get install electric-fence or yum install electric-fence). TODO: Any other features? |
| shared | Build a shared object in addition to the static archive. You probably need a [RPATH](https://wiki.openssl.org/index.php/Compilation_and_Installation#Using_RPATHs) when enabling shared to ensure openssl uses the correct libssl and libcrypto after installation. |
| enable-ec\_nistp\_64\_gcc\_128 | Use on little endian platforms when GCC supports \_\_uint128\_t. ECDH is about 2 to 4 times faster. Not enabled by default because Configure can't determine it. Enable it if your compiler defines \_\_SIZEOF\_INT128\_\_, the CPU is little endian and it tolerates unaligned data access. |
| enable-capieng | Enables the Microsoft CAPI engine on Windows platforms. Used to access the Windows Certificate Store. Also see [Using Windows certificate store through OpenSSL](http://openssl.6102.n7.nabble.com/Using-Windows-certificate-store-through-OpenSSL-td46788.html) on the OpenSSL developer list. |
| no-ssl2 | Disables SSLv2. OPENSSL\_NO\_SSL2 will be defined in the OpenSSL headers. |
| no-ssl3 | Disables SSLv3. OPENSSL\_NO\_SSL3 will be defined in the OpenSSL headers. |
| no-comp | Disables compression independent of zlib. OPENSSL\_NO\_COMP will be defined in the OpenSSL headers. |
| no-idea | Disables IDEA algorithm. Unlike RC5 and MDC2, IDEA is enabled by default |
| no-asm | Disables assembly language routines (and uses C routines) |
| no-dtls | Disables DTLS (useful on mobile devices since carriers often block UDP) |
| no-shared | Disables shared objects (only a static library is created) |
| no-hw | Disables hardware support (useful on mobile devices) |
| no-engines | Disables hardware support (useful on mobile devices) |
| no-threads | Disables threading support. |
| no-dso | Disables the OpenSSL DSO API (the library offers a shared object abstraction layer). If you disable DSO, then you must disable Engines also |
| no-err | Removes all error function names and error reason text to reduce footprint |
| no-npn/no-nextprotoneg | Disables Next Protocol Negotiation (NPN). Use no-nextprotoneg for 1.1.0 and above; and no-npn otherwise |
| no-psk | Disables Preshared Key (PSK). PSK provides mutual authentication independent of trusted authorities, but its rarely offered or used |
| no-srp | Disables Secure Remote Password (SRP). SRP provides mutual authentication independent of trusted authorities, but its rarely offered or used |
| no-ec2m | Used when configuring FIPS Capable Library with a FIPS Object Module that only includes prime curves. That is, use this switch if you use openssl-fips-ecp-2.0.5. |
| no-weak-ssl-ciphers | Disables RC4. Available in OpenSSL 1.1.0 and above. |
| -DXXX | Defines XXX. For example, -DOPENSSL\_NO\_HEARTBEATS. |
| -DOPENSSL\_USE\_IPV6=0 | Disables IPv6. Useful if OpenSSL encounters incorrect or inconsistent platform headers and mistakenly enables IPv6. Must be passed to Configure manually. |
| -L*something*, -l*something*, -K*something*, -Wl,*something* | Linker options, will become part of LDFLAGS. |
| -*anythingelse*, +*anythingelse* | Compiler options, will become part of CFLAGS. |

***Note***: on older OSes, like CentOS 5, BSD 5, and Windows XP or Vista, you will need to configure with no-async when building OpenSSL 1.1.0 and above. The configuration system does not detect lack of the Posix feature on the platforms.

***Note***: you can verify compiler support for \_\_uint128\_t with the following:

# gcc -dM -E - </dev/null | grep \_\_SIZEOF\_INT128\_\_

#define \_\_SIZEOF\_INT128\_\_ 16

[[edit](https://wiki.openssl.org/index.php?title=Compilation_and_Installation&action=edit&section=7)]**PREFIX and OPENSSLDIR**

--prefix and --openssldir control the location of the installed components. The behavior and iterations among --prefix and --openssldir is slightly different between OpenSSL 1.0.2 and below, and OpenSSL 1.1.0 and above.

One thing you should *avoid* is --prefix=/usr when OpenSSL versions are ***not*** [binary compatible](https://wiki.openssl.org/index.php/Binary_Compatibility). You will replace the distro's version of OpenSSL with your version of OpenSSL., and it will most likely break everything, including the package management system.

**OpenSSL 1.0.2 and below**

It is *not* necessary to specify --prefix. If --prefix is *not* specified, then --openssldir is used. If --openssldir is not specified, the the default /usr/local/ssl is used. When it comes time to install, both make install (installs openssl program, headers, libraries and man pages) and make install\_sw (installs openssl program, headers, libraries; but no man pages) both work as expected.

The takeaway is /usr/local/ssl is used by default, and it can be overridden with --openssldir.

**OpenSSL 1.1.0 and above**

OpenSSL 1.1.0 changed the behavior of install rules. You should specify both --prefix and --openssldir to ensure make install (installs openssl program, headers, libraries and man pages) and make install\_sw (installs openssl program, headers, libraries; but no man pages) both work as expected.

The takeaway is /usr/local/ssl is used by default, and it can be overridden with *both* --prefix and --openssldir.

[[edit](https://wiki.openssl.org/index.php?title=Compilation_and_Installation&action=edit&section=8)]**Debug Configuration**

From the list above, its possible to quickly configure a "debug" build with ./config -d. However, you can often get into a more amicable state *without* the [Electric Fence](http://en.wikipedia.org/wiki/Electric_Fence) dependency by issuing:

$ ./config no-asm -g3 -O0 -fno-omit-frame-pointer -fno-inline-functions

Operating system: x86\_64-whatever-linux2

Configuring for linux-x86\_64

Configuring OpenSSL version 1.1.0-pre5-dev (0x0x10100005L)

no-asm [option] OPENSSL\_NO\_ASM

...

Configuring for linux-x86\_64

IsMK1MF =no

CC =gcc

CFLAG =-Wall -O3 -pthread -m64 -DL\_ENDIAN -g3 -O0 -fno-omit-frame-pointer

...

Don't be alarmed about both -O3 and -O0. The last setting *"sticks"*, and that's the -O0.

If you are working in Visual Studio and you can't step into library calls, then see [Step into not working, but can force stepping after some asm steps](http://stackoverflow.com/q/38249235) on Stack Overflow.

[[edit](https://wiki.openssl.org/index.php?title=Compilation_and_Installation&action=edit&section=9)]**Modifying Build Settings**

Sometimes you need to work around OpenSSL's selections for building the library. For example, you might want to use -Os for a mobile device (rather than -O3), or you might want to use the clangcompiler (rather than gcc).

In case like these, its often easier to modify Configure and Makefile.org rather than trying to add targets to the configure scripts. Below is a patch that modifies Configure and Makefile.org for use under the iOS 7.0 SDK (which lacks gcc in /Applications/Xcode.app/Contents/Developer/Toolchains/XcodeDefault.xctoolchain/usr/bin/):

* Modifies Configure to use clang
* Modifies Makefile.org to use clang
* Modifies CFLAG to use -Os
* Modifies MAKEDEPPROG to use $(CC) -M

Setting and resetting of LANG is required on Mac OSX to work around a sed bug or limitation.

OLD\_LANG=$LANG

unset LANG

sed -i "" 's|\"iphoneos-cross\"\,\"llvm-gcc\:-O3|\"iphoneos-cross\"\,\"clang\:-Os|g' Configure

sed -i "" 's/CC= cc/CC= clang/g' Makefile.org

sed -i "" 's/CFLAG= -O/CFLAG= -Os/g' Makefile.org

sed -i "" 's/MAKEDEPPROG=makedepend/MAKEDEPPROG=$(CC) -M/g' Makefile.org

export LANG=$OLD\_LANG

After modification, be sure to dclean and configure again so the new settings are picked up:

make dclean

./config

make depend

make all

...

[[edit](https://wiki.openssl.org/index.php?title=Compilation_and_Installation&action=edit&section=10)]**Using RPATHs**

RPATH's are supported by default on the BSD platforms, but not others. One of the easiest ways to add a RPATH is to perform:

./config -Wl,-rpath=/usr/local/ssl/lib

You can also add and RPATH by hard coding the RPATH into a configure line. For example, on Debian x86\_64 open the file Configure in an editor, copy linux-x86\_64, named it linux-x86\_64-rpath, and make the following change to add the -rpath option. Notice the addition of -Wl,-rpath=... in two places.

"linux-x86\_64-rpath", "gcc:-m64 -DL\_ENDIAN -O3 -Wall -Wl,-rpath=/usr/local/ssl/lib::

-D\_REENTRANT::-Wl,-rpath=/usr/local/ssl/lib -ldl:SIXTY\_FOUR\_BIT\_LONG RC4\_CHUNK DES\_INT DES\_UNROLL:

${x86\_64\_asm}:elf:dlfcn:linux-shared:-fPIC:-m64:.so.\$(SHLIB\_MAJOR).\$(SHLIB\_MINOR):::64",

Above, fields 2 and 6 were changed. They correspond to $cflag and $ldflag in OpenSSL's builds system.

Then, Configure with the new configuration:

$ ./Configure linux-x86\_64-rpath shared no-ssl2 no-ssl3 no-comp \

--openssldir=/usr/local/ssl enable-ec\_nistp\_64\_gcc\_128

Finally, after make, verify the settings stuck:

$ readelf -d ./libssl.so | grep -i rpath

0x000000000000000f (RPATH) Library rpath: [/usr/local/ssl/lib]

$ readelf -d ./libcrypto.so | grep -i rpath

0x000000000000000f (RPATH) Library rpath: [/usr/local/ssl/lib]

$ readelf -d ./apps/openssl | grep -i rpath

0x000000000000000f (RPATH) Library rpath: [/usr/local/ssl/lib]

Once you perform make install, then ldd will produce expected results:

$ ldd /usr/local/ssl/lib/libssl.so

linux-vdso.so.1 => (0x00007ffceff6c000)

ibcrypto.so.1.0.0 => /usr/local/ssl/lib/libcrypto.so.1.0.0 (0x00007ff5eff96000)

...

$ ldd /usr/local/ssl/bin/openssl

linux-vdso.so.1 => (0x00007ffc30d3a000)

libssl.so.1.0.0 => /usr/local/ssl/lib/libssl.so.1.0.0 (0x00007f9e8372e000)

libcrypto.so.1.0.0 => /usr/local/ssl/lib/libcrypto.so.1.0.0 (0x00007f9e832c0000)

...

[[edit](https://wiki.openssl.org/index.php?title=Compilation_and_Installation&action=edit&section=11)]**FIPS Capable Library**

If you want to use FIPS validated cryptography, you download, build and install the FIPS Object Module (openssl-fips-2.0.5.tar.gz) according to the [FIPS User Guide 2.0](https://www.openssl.org/docs/fips/UserGuide-2.0.pdf) and [FIPS 140-2 Security Policy](https://www.openssl.org/docs/fips/SecurityPolicy-2.0.pdf). You then download, build and install the FIPS Capable Library (openssl-1.0.1e.tar.gz).

When configuring the FIPS Capable Library, you must use fips as an option:

./config fips <other options ...>

If you are configuring the FIPS Capable Library with only prime curves (openssl-fips-ecp-2.0.5.tar.gz), then you must configure with no-ec2m:

./config fips no-ec2m <other options ...>

[[edit](https://wiki.openssl.org/index.php?title=Compilation_and_Installation&action=edit&section=12)]**Compile Time Checking**

If you disable an option during configure, you can check if it's available through OPENSSL\_NO\_\* defines. OpenSSL writes the configure options to <openssl/opensslconf.h>. For example, if you want to know if SSLv3 is available, then you would perform the following in your code:

#include <openssl/opensslconf.h>

...

#if !defined(OPENSSL\_NO\_SSL3)

/\* SSLv3 is available \*/

#endif

[[edit](https://wiki.openssl.org/index.php?title=Compilation_and_Installation&action=edit&section=13)]Compilation

After configuring the library, you should run make. If prompted, there's usually no need to make depend since you are building from a clean download.

[[edit](https://wiki.openssl.org/index.php?title=Compilation_and_Installation&action=edit&section=14)]**Quick**

./config <options ...> --openssldir=/usr/local/ssl

make

make test

sudo make install

Various options can be found examining the Configure file (there is a well commented block at its top). OpenSSL ships with SSLv2, SSLv3 and Compression enabled by default (see my $disabled), so you might want to use no-ssl2 no-ssl3, no-ssl3, and no-comp.

[[edit](https://wiki.openssl.org/index.php?title=Compilation_and_Installation&action=edit&section=15)]Platfom specific

[[edit](https://wiki.openssl.org/index.php?title=Compilation_and_Installation&action=edit&section=16)]**Linux**

[[edit](https://wiki.openssl.org/index.php?title=Compilation_and_Installation&action=edit&section=17)]**Intel**

[[edit](https://wiki.openssl.org/index.php?title=Compilation_and_Installation&action=edit&section=18)]**ARM**

[[edit](https://wiki.openssl.org/index.php?title=Compilation_and_Installation&action=edit&section=19)]**X32 (ILP32)**

X32 uses the 32-bit data model (ILP32) on x86\_64/amd64. To properly configure for X32 under current OpenSSL distributions, you must use Configure and use the x32 triplet:

# ./Configure LIST | grep x32

linux-x32

...

Then:

# ./Configure linux-x32

Configuring OpenSSL version 1.1.0-pre6-dev (0x0x10100006L)

no-asan [default] OPENSSL\_NO\_ASAN (skip dir)

...

Configuring for linux-x32

CC =gcc

CFLAG =-Wall -O3 -pthread -mx32 -DL\_ENDIAN

SHARED\_CFLAG =-fPIC

...

If using an amd64-compatible processor and GCC with that supports \_\_uint128\_t, then you usually add enable-ec\_nistp\_64\_gcc\_128 in addition to your other flags.

[[edit](https://wiki.openssl.org/index.php?title=Compilation_and_Installation&action=edit&section=20)]**Windows**

3noch wrote a VERY good guide [here](http://developer.covenanteyes.com/building-openssl-for-visual-studio/). Like he said in his article, make absolutely sure to create separate directories for 32 and 64 bit versions.

[[edit](https://wiki.openssl.org/index.php?title=Compilation_and_Installation&action=edit&section=21)]**W32 / Windows NT - Windows 9x**

type INSTALL.W32

* you need Perl for Win32. Unless you will build on Cygwin, you will need ActiveState Perl, available from <http://www.activestate.com/ActivePerl>.
* one of the following C compilers:
  + Visual C++
  + Borland C
  + GNU C (Cygwin or MinGW)
* Netwide Assembler, a.k.a. NASM, available from <http://nasm.sourceforge.net/> is required if you intend to utilize assembler modules. Note that NASM is now the only supported assembler.

[[edit](https://wiki.openssl.org/index.php?title=Compilation_and_Installation&action=edit&section=22)]**W64**

Read first the INSTALL.W64 documentation note containing some specific 64bits information. See also INSTALL.W32 that still provides additonnal build information common to both the 64 and 32 bit versions.

You may be surprised: the 64bit artefacts are indeed output in the out32\* sub-directories and bear names ending \*32.dll. Fact is the 64 bit compile target is so far an incremental change over the legacy 32bit windows target. Numerous compile flags are still labelled "32" although those do apply to both 32 and 64bit targets.

The important pre-requisites are to have PERL available (for essential file processing so as to prepare sources and scripts for the target OS) and of course a C compiler like Microsoft Visual Studio for C/C++.

Using MS Visual Studio:

1. launch a Visual Studio tool x64 Cross Tools Command prompt
2. change to the directory where you have copied openssl sources cd c:\myPath\openssl
3. configure for the target OS with the command perl Configure VC-WIN64A. You may also be interested to set more configuration options as documented in the general INSTALL note (for UNIX targets). For instance: perl Configure no-asm VC-WIN64A.
4. prepare the target environment with the command: ms\do\_win64a
5. ensure you start afresh and notably without linkable products from a previous 32bit compile (as 32 and 64 bits compiling still share common directories) with the command: nmake -f ms\ntdll.mak clean for the DLL target and nmake -f ms\nt.mak clean for static libraries.
6. build the code with: nmake -f ms\ntdll.mak (respectively nmake -f ms\nt.mak )
7. the artefacts will be found in sub directories out32dll and out32dll.dbg (respectively out32 and out32.dbg for static libraries). The libcrypto and ssl libraries are still named libeay32.lib and ssleay32.lib, and associated includes in inc32 ! You may check this is true 64bit code using the Visual Studio tool 'dumpbin'. For instance dumpbin /headers out32dll/libeay32.lib | more, and look at the FILE HEADER section.
8. test the code using the various \*test.exe programs in out32dll. Use the 'test' make target to run all tests as in nmake -f ms\ntdll.mak test
9. we recommend that you move/copy needed includes and libraries from the "32" directories under a new explicit directory tree for 64bit applications from where you will import and link your target applications, similar to that explained in INSTALL.W32.

[[edit](https://wiki.openssl.org/index.php?title=Compilation_and_Installation&action=edit&section=23)]**Windows CE**

[[edit](https://wiki.openssl.org/index.php?title=Compilation_and_Installation&action=edit&section=24)]**OS X**

The earlier discussion presented a lot of information (and some of it had OS X information). Here are the TLDR versions to configure, build and install the library.

If configuring for 64-bit OS X, then use a command similar to:

./Configure darwin64-x86\_64-cc shared enable-ec\_nistp\_64\_gcc\_128 no-ssl2 no-ssl3 no-comp --openssldir=/usr/local/ssl/macos-x86\_64

make depend

sudo make install

If configuring for 32-bit OS X, then use a command similar to:

./Configure darwin-i386-cc shared no-ssl2 no-ssl3 no-comp --openssldir=/usr/local/ssl/macosx-i386

make depend

sudo make install

If you want to build a multiarch OpenSSL library, then see this answer on Stack Overflow: [Build Multiarch OpenSSL on OS X](http://stackoverflow.com/a/25531033/608639).

[[edit](https://wiki.openssl.org/index.php?title=Compilation_and_Installation&action=edit&section=25)]**iOS**

[[edit](https://wiki.openssl.org/index.php?title=Compilation_and_Installation&action=edit&section=26)]**Android**

Visit [Android](https://wiki.openssl.org/index.php/Android) and [FIPS Library and Android](https://wiki.openssl.org/index.php/FIPS_Library_and_Android).

[[edit](https://wiki.openssl.org/index.php?title=Compilation_and_Installation&action=edit&section=27)]**More**

[[edit](https://wiki.openssl.org/index.php?title=Compilation_and_Installation&action=edit&section=28)]**VAX/VMS**

I you wonder what are files ending with .com like test/testca.com those are VAX/VMX scripts. This code is still maintained.

[[edit](https://wiki.openssl.org/index.php?title=Compilation_and_Installation&action=edit&section=29)]**OS/2**

[[edit](https://wiki.openssl.org/index.php?title=Compilation_and_Installation&action=edit&section=30)]**NetWare**

5.x 6.x

[[edit](https://wiki.openssl.org/index.php?title=Compilation_and_Installation&action=edit&section=31)]**HP-UX**

[HP-UX Itanium FIPS and OpenSSL build](https://wiki.openssl.org/index.php/HP-UX_Itanium_FIPS_and_OpenSSL_build)

[[edit](https://wiki.openssl.org/index.php?title=Compilation_and_Installation&action=edit&section=32)]Autoconf

OpenSSL uses its own configuration system, and does not use Autoconf. However, a number of popular projects use both OpenSSL and Autoconf, and it would be usful to detect either OPENSSL\_init\_ssl andOPENSSL\_init\_ssl from libssl. To craft a feature test for OpenSSL that recognizes both OPENSSL\_init\_ssl and OPENSSL\_init\_ssl, you can use the following.

if test "$with\_openssl" = yes ; then

dnl Order matters!

if test "$PORTNAME" != "win32"; then

AC\_CHECK\_LIB(crypto, CRYPTO\_new\_ex\_data, [], [AC\_MSG\_ERROR([library 'crypto' is required for OpenSSL])])

FOUND\_SSL\_LIB="no"

AC\_CHECK\_LIB(ssl, OPENSSL\_init\_ssl, [FOUND\_SSL\_LIB="yes"])

AC\_CHECK\_LIB(ssl, SSL\_library\_init, [FOUND\_SSL\_LIB="yes"])

AS\_IF([test "x$FOUND\_SSL\_LIB" = xno], [AC\_MSG\_ERROR([library 'ssl' is required for OpenSSL])])

else

AC\_SEARCH\_LIBS(CRYPTO\_new\_ex\_data, eay32 crypto, [], [AC\_MSG\_ERROR([library 'eay32' or 'crypto' is required for OpenSSL])])

FOUND\_SSL\_LIB="no"

AC\_SEARCH\_LIBS(OPENSSL\_init\_ssl, ssleay32 ssl, [FOUND\_SSL\_LIB="yes"])

AC\_SEARCH\_LIBS(SSL\_library\_init, ssleay32 ssl, [FOUND\_SSL\_LIB="yes"])

AS\_IF([test "x$FOUND\_SSL\_LIB" = xno], [AC\_MSG\_ERROR([library 'ssleay32' or 'ssl' is required for OpenSSL])])

fi

fi

Many thanks to the Postgres folks for donating part of their configure.in. Also see [How to tell Autoconf “require symbol A or B” from LIB?](http://stackoverflow.com/q/39285733) on Stack Overflow.

[Categories](https://wiki.openssl.org/index.php/Special:Categories):

* [Shell level](https://wiki.openssl.org/index.php/Category:Shell_level)
* [Installation](https://wiki.openssl.org/index.php/Category:Installation)
* [Compilation](https://wiki.openssl.org/index.php/Category:Compilation)

# 参考

<http://www.linuxidc.com/Linux/2014-10/108502.htm>

OpenSSL在Windows7 64bit + vs2010中win32的编译步骤：

1. 从https://www.openssl.org/source/下载OpenSSL源代码 openssl-1.0.1g版本；

2. 从http://www.activestate.com/activeperl/downloads下载ActivePerl5.16.3 x64(ActivePerl为perl的一个脚本解释器)；

3. 将ActivePerl安装到D:\ProgramFiles\Perl64目录下，打开命令提示符，将其定位到D:\ProgramFiles\Perl64\eg 目录下，执行perl example.pl ，若提示Hello fromActivePerl! 则说明Perl安装成功，可以开始使用Perl的相关命令来进行OpenSSL的安装了；

4. 将openssl-1.0.1g解压缩到E:\OpenSSL\openssl-1.0.1g目录下；

5. 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. 将命令提示符定位到E:\OpenSSL\openssl-1.0.1g；

7. 输入perl Configure VC-WIN32 --perfix=E:\OpenSSL\openssl (将其安装到E:\OpenSSL\openssl)；

8. 输入ms\do\_nasm ；

9. 将命令提示符定位到C:\Program Files (x86)\Microsoft Visual Studio 10.0\VC\bin ， 然后输入 vcvars32.bat，会显示Settingenvironment for using Microsoft Visual Studio 2010 x86 tools. 如果没有这一步，会提示nmake不是内部或外部命令；

10. 再次将命令提示符定位到E:\OpenSSL\openssl-1.0.1g；

11. 输入 nmake –f ms\ntdll.mak ；执行完后会在openssl-1.0.1g目录下生成一个out32dll文件夹，里面包含了一些静态库、动态库和.exe文件；

12. 输入 nmake -f ms\ntdll.mak test ；若最终显示passed all tests说明生成的库正确；

13. 输入 nmake -f ms\ntdll.mak install ；则会在E:\OpenSSL\openssl目录下生成bin、include、lib、ssl四个文件夹；

14. 以上编译的是release库，若编译debug库，则将以上第7步中的VC-WIN32改成debug-VC-WIN32即可；

15. 若编译静态库则用ms\nt.mak替换掉上面用到的ms\ntdll.mak即可；

16. 若生成不带汇编支持的库，则需将以上第7、8步用perl Configure VC-WIN32 no-asm --prefix=E:\OpenSSL\openssl 和 ms\do\_ms替换即可；

17. 在E:\OpenSSL\openssl-1.0.1g\tmp32dll文件夹下包含相应的汇编文件。