Building Ice for C++ on Linux

This page describes the Ice source distribution, including information about compiler requirements, third-party dependencies, and instructions for building and testing the distribution. If you prefer, you can download binary packages for supported platforms that contain pre-compiled libraries, executables, and everything else necessary to build Ice applications on Linux.

On this page:

C++ Build Requirements for Linux

Ice is expected to build and run properly on any recent Linux distribution for x86 and x86_64, and was extensively tested using the operating systems and compiler versions listed on our platforms page.

Third-party Libraries

Ice has dependencies on a number of third-party libraries:

- Berkeley DB 5.3
- expat 2.0
- OpenSSL 0.9.8 or later
- bzip2 1.0
- mcpp 2.7.2 (with patches)

Some of these packages may have been included in your Linux distribution. For those packages that are not installed or have an older version than what is listed above, we recommend downloading the Ice third-party source archive. This archive contains the source distributions for each of the third-party dependencies, as well as required source patches and configuration instructions.

ZeroC also supplies RPMs at the link above for Berkeley DB on RHEL 6, AMZN and SLES 11, as well as packages for Ubuntu 13.04.

Compiling and Testing Ice for C++ on Linux

Extract the Ice archive in any directory you like (for example, in your home directory):

```
$ tar xvfz Ice-3.5.1.tar.gz
```

Change the working directory to `Ice-3.5.1/cpp`:

```
$ cd Ice-3.5.1/cpp
```

Edit `config/Make.rules` to establish your build configuration. The comments in the file provide more information. Pay particular attention to the variables that define the locations of the third-party libraries.

Now you're ready to build Ice:

```
$ make
```

This will build the Ice core libraries, services, tests and examples.

Python is required to run the test suite. After a successful build, you can run the tests as follows:

```
$ make test
```

This command is equivalent to:

```
$ python allTests.py
```

If everything worked out, you should see lots of "ok" messages. In case of a failure, the tests abort with "failed".

If you want to try out any of the demos, make sure to update your `PATH` environment variable to add the `bin` directory, and your `LD_LIBRARY_PATH` environment variable to add the `lib` directory:

```
$ export PATH=`pwd`/bin:$PATH
$ export LD_LIBRARY_PATH=`pwd`/lib:$LD_LIBRARY_PATH
```

64-bit Source Builds on Linux x86_64

To build Ice in 64-bit mode, you need to do the following:

- Obtain or build all the third-party dependencies, and put the 64-bit libraries in the `lib64` directories. For example, put Berkeley DB 64-bit libraries in `$DB_HOME/lib64`.
- Build and test as described above (with gcc).

32-bit Source Builds on Linux x86_64

By default, builds on x86_64 are 64-bit. To perform a 32-bit build on a x86_64 Linux system, set the environment variable `LP64` to no, as shown below:

```
$ export LP64=no
```

Installing a C++ Source Build on Linux

Simply run `make install`. This will install Ice in the directory specified by the `prefix` variable in `config/Make.rules`.

After installation, make sure that the *`prefix`*`/bin` directory is in your `PATH`.

If you choose to not embed a runpath into executables at build time (see your build settings in `cpp/config/Make.rules`) or did not create a symbolic link from the runpath directory to the installation directory, you also need to add the *`prefix`*`/lib` directory to your `LD_LIBRARY_PATH`.

When compiling Ice programs, you must pass the location of the *`prefix`*`/include` directory to the compiler with the `-I` option, and the location of the *`prefix`*`/lib` directory with the `-L` option.

On an x86_64 system, the libraries are installed in *`prefix`*`/lib64` unless `LP64` is set to no. No other changes are necessary.