

# Compiling mauveAligner from source

Although Mauve is provided as a pre-compiled binary for Windows, Linux, and Mac OS X, we also make the source code available so that users can modify Mauve and compile it on other platforms. This chapter describes the procedure for compiling the command-line mauveAligner tool from source code on a unix system. Project files for Microsoft Visual studio are included to compile Mauve in a Windows environment. CodeWarrior project files have been obsoleted.

## 0. Prerequisites

mauveAligner depends on several additional software packages. The latest code in our subversion repository gets packaged automatically every night and is available from <http://gel.ahabs.wisc.edu/mauve/source/snapshots>. The developers make no guarantee that source snapshots will compile or otherwise work.

libGenome <http://gel.ahabs.wisc.edu/mauve/source> libMems <http://gel.ahabs.wisc.edu/mauve/source> mauveAligner  
<http://gel.ahabs.wisc.edu/mauve/source> libMuscle <http://gel.ahabs.wisc.edu/mauve/source> pkg-config  
<http://www.freedesktop.org/software/pkgconfig/releases/> boost <http://www.boost.org>

Each of these packages must be downloaded and installed in order to compile mauveAligner successfully. Alternatively, it may be preferable to check the source code out directly from the sourceforge subversion repository, especially if the code will be modified and changes will be committed back to the repository. To do so, execute the following series of commands inside a development directory:

```
svn co https://mauve.svn.sourceforge.net/svnroot/mauve/libGenome/trunk libGenome
svn co https://mauve.svn.sourceforge.net/svnroot/mauve/libClustalW/trunk libClustalW
svn co https://mauve.svn.sourceforge.net/svnroot/mauve/libMems/trunk libMems
svn co https://mauve.svn.sourceforge.net/svnroot/mauve/muscle/trunk muscle
svn co https://mauve.svn.sourceforge.net/svnroot/mauve/sgEvolver/trunk sgEvolver
svn co https://mauve.svn.sourceforge.net/svnroot/mauve/mauveAligner/trunk mauveAligner
```

## 1. Installing Boost

Boost may be installed as a pre-compiled RPM on Linux systems, or from source on Windows and Mac OS X. We recommend building boost from source on all platforms. A default build of the boost source may be done using the commands:

```
export NO_COMPRESSION="true"
bjam --prefix=$HOME release link=static --with-filesystem --with-iostreams --with-program_options -d2 install
```

## 2. Installing libGenome

Once downloaded and untarred, libGenome can be compiled and installed using the following commands from within the libGenome directory:

```
./configure --prefix=$HOME
make
make install
```

libGenome is installed by default in the /usr/local directory. The --prefix argument to ./configure is optional and specifies the location where libGenome gets installed.

## 3. Installing muscle

The installation procedure for the muscle library is similar to that for libGenome. From the software package's directory, execute the following commands:

```
./configure --prefix=$HOME
make
make install
```

An error usually occurs after running the 'make' command. It is safe to ignore and simply continue with 'make install'. Again, the --prefix is an optional argument to ./configure which specifies the location where the library gets installed.

## 4. Installing libMems

Since libMems depends on the functionality provided by libGenome, muscle, and boost these packages must be installed before libMems. Also, libMems requires the pkg-config software to determine the location of libGenome and libClustalW during the installation process. From the libMems directory, execute the following commands:

```
./configure --prefix=$HOME
make
make install
```

Also, if muscle or libGenome were installed to a non-standard directory, the directory may need to be added to the PKG\_CONFIG\_PATH environment variable. For example, if libGenome were installed in the prefix \$HOME, the PKG\_CONFIG\_PATH variable could be set with this command: `export PKG_CONFIG_PATH="$PKG_CONFIG_PATH:$HOME/lib/pkgconfig"` It's a good idea to add PKG\_CONFIG\_PATH to the .profile or .bashrc or another login script so it doesn't need to be set manually every time the user logs in. As with the other packages, the installation prefix can be specified using the --prefix argument.

## 5. Compiling mauveAligner

Once libGenome, muscle, and libMems are installed, the aligner software can be compiled. From the mauveAligner directory, issue the following commands:

```
./configure
make
make install
```

In addition to mauveAligner and progressiveMauve, several supporting applications will be compiled. We do not have explicit documentation for them. Many of them are self-explanatory and give their usage instructions when run without arguments.

The statically linked version of mauveAligner is called mauveStatic. It includes the necessary part of the support libraries directly in the program and can be used on other systems without first installing the support libraries. Note that Mac OS X doesn't support static linking and any applications built on OS X will be dynamically linked.

## Compiling from a source snapshot

The latest development snapshots do not have a complete build system. In order to compile snapshots the build system must be regenerated using a recent version of the autotools software. To prepare a source snapshot for a build, execute the following command:

```
./autogen.sh
```

You will need GNU autotools installed, including autoconf and automake. The source directory will now be ready for a build with the usual configure, make, make install procedure.

## Other notes

Since Apple's migration to Intel CPUs, Mac software should be compiled for both Intel and PowerPC CPUs, at least for the time being. Apple refers to programs with both PPC and x86 code as "universal". Building a universal library with GNU autotools is nearly impossible, so it's necessary to build x86 and PPC libraries and binaries separately and then join the result with the lipo tool. To build for a PowerPC CPU on an Intel machine, set the following environment variables:

```
export CFLAGS="-O3 -g -isysroot /Developer/SDKs/MacOSX10.3.9.sdk -arch ppc"
export CXXFLAGS="-O3 -g -isysroot /Developer/SDKs/MacOSX10.3.9.sdk -arch ppc"
```

And compile boost with `bjam "-sTOOLS=darwin" "-sGXX=g++ -O3 -g -isysroot /Developer/SDKs/MacOSX10.3.9.sdk -arch ppc" "-sGCC=gcc -O3 -g -isysroot /Developer/SDKs/MacOSX10.3.9.sdk -arch ppc" "-sBUILD=release <linkflags>-Wl,-syslibroot,/Developer/SDKs/MacOSX10.3.9.sdk <runtime-link>static <threading>single/multi" --prefix=$HOME` install The rest of the build procedure remains the same. The resulting libraries and applications will run on any PowerPC mac with an OS as old as 10.3.9. To create a single universal binary for mauveAligner, run something akin to the following command `lipo -create mauveAligner-intel mauveAligner-ppc -output mauveAligner` where mauveAligner-intel and mauveAligner-ppc are the paths to the intel and ppc mauveAligner binaries, respectively.

## A complete example

The following series of commands will build all libraries and source code from the latest source snapshots on an x86 Linux system, installing software to the user's home directory:

```
# create essential directories if they don't exist
mkdir -p ~/bin
mkdir -p ~/lib
export PATH="$HOME/bin:$PATH"
export LD_LIBRARY_PATH="$HOME/lib:$LD_LIBRARY_PATH"

# make a build directory
mkdir build
cd build

# download the source
wget http://pkgconfig.freedesktop.org/releases/pkg-config-0.20.tar.gz
wget http://internap.dl.sourceforge.net/sourceforge/boost/boost_1_33_1.tar.bz2
wget http://internap.dl.sourceforge.net/source
forge/boost/boost-jam-3.1.11-1-linuxx86.tgz
wget http://gel.ahabs.wisc.edu/mauve/source/snapshots/libgenome-snapshot.tar.gz
wget http://gel.ahabs.wisc.edu/mauve/source/snapshots/libclustalw-snapshot.tar.gz
wget http://gel.ahabs.wisc.edu/mauve/source/snapshots/libmems-snapshot.tar.gz
wget http://gel.ahabs.wisc.edu/mauve/source/snapshots/muscle-snapshot.tar.gz
wget http://gel.ahabs.wisc.edu/mauve/source/snapshots/mauvealigner-snapshot.tar.gz
wget http://gel.ahabs.wisc.edu/mauve/source/snapshots/sgEvolver-snapshot.tar.gz

# build pkg-config
tar xzf pkg-config-0.20.tar.gz
cd pkg-config-0.20
./configure --prefix=$HOME
make install
export PKG_CONFIG_PATH="$HOME/lib/pkgconfig"
cd ..

# build boost
tar xzf boost-jam-3.1.11-1-linuxx86.tgz
cp cp boost-jam-3.1.13-1-linuxx86/bjam ~/bin/
tar xjf boost_1_33_1.tar.bz2
cd boost_1_33_1
bjam --prefix=$HOME install
cd ..

# build libraries
cd libGenome
./autogen.sh
./configure --prefix=$HOME && make install
cd ..

cd muscle
./autogen.sh
./configure --prefix=$HOME && make ; make ; make install
cd ..

cd libMems
./autogen.sh
./configure --prefix=$HOME --with-boost=$HOME && make install
cd ..

cd mauveAligner
./autogen.sh
```

```
./configure --prefix=$HOME && make install  
cd ..
```

```
cd sgEvolver  
./autogen.sh  
./configure --prefix=$HOME && make install  
cd ..
```

If you will be building the source more than once, or especially if you will be editing and recompiling the source, it will be desirable to have PATH, LD\_LIBRARY\_PATH, and PKG\_CONFIG\_PATH set automatically on login. Edit one of \$HOME/.profile or \$HOME/.bashrc or \$HOME/.profile.local to set the environment variables.

To build the GUI, execute the following additional commands:

```
wget http://gel.ahabs.wisc.edu/mauve/source/snapshots/mauve-snapshot.tar.gz  
tar xzf mauve-snapshot.tar.gz  
cp mauveAligner/src/mauveStatic mauve/linux-x86/mauveAligner  
cp mauveAligner/src/progressiveMauveStatic mauve/linux-x86/progressiveMauve  
cp muscle/muscle mauve/linux-x86/muscle_aedcd mauve  
ant dist
```

The Mauve GUI build will reside in mauve/dist/