Building Lilium Voyager

For the source files of Lilium Voyager please use these sources. The git URL is `https://github.com/zturtleman/lilium-voyager.git`.

Also note, that building for Windows x86 does not result in a build that works on Windows 9x/ME any more. You need at least XP or Windows 2000 for that to work. (But then again, if you are concerned with security issues for needing to use a more modern client, you should not use those systems either any longer.)

Linux

On Linux it is possible to build for both variations, x86 and amd64. It is also possible to cross compile for Windows (32 and 64 bit), see next section for that. To compile on Linux you require the following prerequisites installed:

- `git` (versioning tool) and
- `libsdl2-dev` (SDL 2 development package).

If you want to compile the 32 bit version on an amd64 platform, you might also need to do these steps:

1. `sudo dpkg --add-architecture i386`: Activate i386 packages as well,
2. `sudo apt-get install libc6-dev-i386 libsdl2-dev:i386`: Install the SDL and libc6 for i386.

Once you have installed these packages, you can start with cloning the git repository on the console, containing the EF sources, like this:

```
git clone https://github.com/zturtleman/lilium-voyager.git
```

It will download the new source files to a newly created folder `lilium-voyager`. Switch to it and run `make`. This will build EF for your own platform. The resulting binaries will be placed inside the subfolder `build`, e. g. for an AMD 64 architecture the folder `release-linux-x86_64` will be created. From there you only need the following four files:

- `liliumvoyded.x86_64` (dedicated server binary, used to set up a server),
- `liliumvoyhm_renderer_opengl1_x86_64.so` (graphics library),
- `liliumvoyhm_renderer_opengl2_x86_64.so` (graphics library),
- `liliumvoyhm.x86_64` (client binary, used to play on a server).

If you want to update an already existing Lilium Voyager folder, just run `git pull` for updating the source repository and run `make clean` (removing files created during earlier run) and afterwards `make` to run the compilation process again.

If you want to create binaries for another architecture than you own (e. g. compile x86 binaries on an amd64 system), you can specify this with the ARCH property. For the former example the command would be

`make ARCH=x86`.

So what `make` actually does on an amd64 system is the same as `make ARCH=x86_64`. Similarly, on an ARM system the *make* command does the same as `make ARCH=arm`.

Cross compile for Windows via Linux

For cross-compiling from Linux to Windows the MingW environment is necessary. Use something like `sudo apt-get install mingw-w64` for Linux (here: for Debian & derived distributions...) to install it. Then run the compilation process like this:

`make PLATFORM=mingw32 ARCH=x86_64` (64 bit) or

`make PLATFORM=mingw32 ARCH=x86` (32 bit).

MacOS

You will need to run the bash on MacOS and also install git. Check out the source files from the git repository as described for Linux. Then run the following command:

`./make-macosx.sh x86_64`.

As MacOS today only comes as 64 bit version, this should be all you need. From the results in the build directory, you will only need the following elements:

- `Lilium Voyager.app` (Quick starter for the game) or
- `liliumvoyded.x86_64` (for setting up a dedicated server),
- `liliumvoyhm_renderer_opengl1_x86_64.dylib` (graphics library),
- `liliumvoyhm_renderer_opengl2_x86_64.dylib` (graphics library),
- `liliumvoyhm.x86_64` (game binary) and
- `libSDL2-2.0.0.dylib` (SDL library).

If you're running Mac OS 10.6.x and have the Mac OS 10.5 developer SDK you can make a universal binary for ppc, x86, and x86_64 that is compatible with OS X 10.5 or later Mac OS versions.

- Remove the build directory if you previously built ioq3 using make, because it defaults to 10.7 compatibility.
- Run `./make-macosx-ub.sh`

If you're running Mac OS 10.7 or newer you can build a x86 / x86_64 universal binary that is compatible with 10.7 or later. (Not particularly useful since all Mac OS 10.7 and later are x86_64.)

- Run `make ARCH=x86` or
- `make ARCH=x86_64` and
- run `./make-macosx-app.sh release`.

Windows - msys2

64-Bit Binaries

To build 64-bit binaries, follow these instructions:

1. Install msys2 from https://msys2.github.io/, following the instructions there. It doesn't matter which version you download, just get one appropriate for your OS.
2. Start “MSYS2 MinGW 64-bit” from the Start Menu.
3. Install mingw-w64-x86_64: `pacman -S mingw-w64-x86_64-gcc`
4. Install make: `pacman -S make`
5. Grab latest ioq3 source code from github. Use git, or just grab https://github.com/ioquake/ioq3/archive/master.zip and unzip it somewhere.
6. Change directory to where you put the source and run make. Note that in msys2, your drives are linked as folders in the root directory: C:\ is /c/, D:\ is /d/, and so on:

```
cd /c/ioq3
make ARCH=x86_64
```

1. Find the executables and dlls in `build/release-mingw64-x86_64`.

32-Bit Binaries

To build 32-bit binaries, follow these instructions:

1. Install msys2 from https://msys2.github.io/, following the instructions there. It doesn't matter which version you download, just get one appropriate for your OS.
2. Start “MSYS2 MinGW 32-bit” from the Start Menu.
3. Install mingw-w64-i686-gcc: `pacman -S mingw-w64-i686-gcc`
4. Install make from msys: `pacman -S make`
5. Grab latest ioq3 source code from github Use git, or just grab https://github.com/ioquake/ioq3/archive/master.zip and unzip it somewhere.
6. Change directory to where you put the source and run make. Note that in msys2, your drives are linked as folders in the root directory: C:\ is /c/, D:\ is /d/, and so on.

```
cd /c/ioq3
make ARCH=x86 WINDRES="windres -F pe-i386"
```

1. Find the executables and dlls in build/release-mingw32-x86 .

Windows - cygwin

1. Install Cygwin

Download the Cygwin setup package from http://cygwin.com/install.html.

Choose either the 32-bit or 64-bit environment. 32-bit will work fine on both 32 and 64 bit versions of Windows. The setup program is also your Cygwin environment updater. If you have an existing Cygwin environment, the setup program will, by default, update your existing packages.

Choose where you want to install Cygwin. The entire environment is self-contained in it's own folder, but you can also interact with files from outside the environment if you want to as well. The default install path is `C:\Cygwin`.

1. Choose a mirror to download packages from, such as the kernel.org mirrors.
2. Choose a “storage area” for your package downloads.

2. Package selection

The next screen you see will be the package selections screen. In the upper left is a search box. This is where you will want to search for the necessary packages.

These are the package names you'll want to search for:

- `mingw64-i686-gcc-core` (for building 32bit binaries),
- `mingw64-i686-gcc-g++` (also for 32bit… C++ support… not required for ioquake3, but useful for compiling other software),
- `mingw64-x86_64-gcc-core` (for building 64bit binaries),
- `mingw64-x86_64-gcc-g++` (for 64bit, same as above),
- `make`,
- `bison` and
- `git`.

When you search for your packages you'll see category listings. These packages would all be under the “Devel” category.

To select a package, change the 'Skip' to the version of the package you want to install. The first click will be the newest version and subsequent clicks will allow you to choose older versions of the package. In our case here, you're probably good choosing the latest and greatest.

3. Install packages

After you have selected your packages, just hit 'Next' in the lower right. Cygwin will automatically add package dependencies. Hit next again to let the install run. Done.

The entire environment uses about 1GB of disk space (as opposed to about 6GB for a Visual Studio install).

4. Using Cygwin to check out and build ioquake3

After the install has completed you should have a 'Cygwin Terminal' icon on your Desktop. This is the bash shell for Cygwin, so go ahead and run it.

At the command prompt type: `git clone https://github.com/zturtleman/lilium-voyager.git`

This will download the Lilium Voyager master branch source into a new `lilium-voyager` folder.

Change to the folder that was created: `cd lilium-voyager`

You can build EF for 32 or 64 bit Windows by running one of the following

- `make ARCH=x86` (32 bit) or

- `make ARCH=x86_64` (64 bit).

You can also compile for Linux, when adding `PLATFORM=Linux`.

After the build completes the output files will be in the `build` folder. The path would be `C:\Cygwin\home\<username>\lilium-voyager\build\release-mingw32-arch` for the default Cygwin install.

Pre-Compiled Binaries

As many users would find it requiring too much effort compiling Lilium Voyager themselves, pre-compiled binaries can be found here. These are builds based directly upon the Github repository, compiled by 7Saturn.

[Back to Star Trek - Voyager Elite Force] [Back to the games database]