Wired How To's

Compile Software from Source Code

Most of the time, especially in the Windows and Mac OS X world, we end users don't have to compile our own software. The programmers of the software wrote the code, and then compiled it into a "binary executable" that's designed to run on our particular type of computer and operating system.

Increasingly, though, with the popularity of open-source software, where the source code is available for one and all to read and enjoy, the final step of compiling the code into a runnable program is left up to the user. It needs to be compiled specifically for the type of system it'll run on, and maintaining separate binaries for Windows XP, Windows Vista, Mac OS X Tiger, Panther, Linux x86, and all the rest is a bother. Also, publishing just the code leaves users free to modify it, often with publicly available patches, before compiling, to customize it to their needs.

If you find yourself having to (or wanting to) compile a piece of software, the process is fairly simple. Every piece of software is different, but here is a summary of the most common scenario. First, you'll need a command line and a compiler.

Linux
You almost certainly already have a command line, and a compiler called gcc. Try typing `gcc` on the command line. If it tells you `command not found`, install gcc using your package manager.

Mac OS X
Your command line is Terminal, which is located in Applications/Utilities. You'll need to download XCode, which contains Apple's compiler, from the Apple developer tools web site.

Windows
Microsoft provides a compiler with its free downloadable Visual Studio Express, but Cygwin is both more useful and more user-friendly. Download and install it, making sure to include all the packages from the "Devel" section when you choose packages.

Downloading
Practically all source packages come in a zipped-up archive file. This will have the suffix .tgz or .tar.gz. It will also have a name and a version number, something like example-3.2.8.tar.gz. You should make yourself a directory to work in, called "source" or "build" or something along those lines. Using a browser, download the source file into that directory.

Unpacking
From your command line, go into your working directory (using the `cd` command):

```
cd source
```

The quickest way to unzip the archive is tar:

```
tar -xzvf example-3.2.8.tar.gz
```

This will unzip all the files of the source code into a new subdirectory with the same name as your application, including the version numner. Go into this directory by typing:

```
cd example-3.2.8
```

Reading the Documentation
Practically every source package contains some reading material, typically files with names in all caps, like README and INSTALL. Read these! They tell you how to proceed. There may also be documentation for your specific situation, like README.macosx. You can use the `less` command on the command line to read them:

```
less INSTALL
```

or just open them in your favorite text editor. The documentation may point you to other software that you need to install before you can install this package ("dependencies"), or to quirks of the installation process that you need to be alert for. Much of this may be covered on the software's web page as well.

Building
The process may differ, but the most common procedure is as follows. Type:

```
./configure
```

The dot-slash beforehand means you want to run the configuration tool in the current directory. This runs some diagnostic tests on your computer, figuring out whether the software has everything it needs and where important files are. You might have to specify the location of certain files on your computer if they're not in the obvious place -- the documentation should cover this; e.g.:

```
./configure --ssl-dir=/usr/local/include
```

For a full list of all the options you can give to the configure tool, run:

```
./configure --help
```

The configuration process may take several minutes. When it's done, if it doesn't give you an error message, you're ready to compile. If it does give you an error message, refer to the Troubleshooting section below. To compile the software, just type:

```
make
```

This performs the meat of the operation. If all goes well, it should take a while, and may monopolize your computer's processor while it's running. Don't worry, compiling software is intensive work. When it ends, if you still don't see an error message, you're ready for the last step. Note that you probably won't see a congratulations message either, but if there's no error, you've succeeded. The software has been compiled. All that remains to do is to put it where it belongs.

```
make install
```

will place the various files that have been built in their proper locations in the filesystem. Now they're ready to be used.

Troubleshooting
If any of the steps above don't go smoothly, there are a few systematic steps to take that will help figure out what the problem is. Make sure you've followed all the instructions rigorously, and that you have any necessary dependencies installed.

If you can't figure out where you went wrong, search for the error message you received, on Google, Usenet, and in the forums and/or mailing list archives for the software. If that fails, you can probably e-mail the mailing list with your question, or even contact the software's author directly if there's no active mailing list. Either way, be sure you've done your homework first. Here is a good tutorial on how to ask for help.

Created by mike@hidden on Jul 19 5:12pm. Updated by mike@hidden on Jul 20 3:25pm.