# Contributing to FontForge

## Reporting An Issue

See [When Things Go Wrong With FontForge Itself](http://designwithfontforge.com/en-US/When\_Things\_Go\_Wrong\_With\_Fontforge\_Itself.html)

## Contributing Code

Merging into the master branch requires review and approval of the pull request

by an active core contributor to the project. The reviewer must fully

comprehend the code being modified, and any changes must be strictly

non-regressive and non-breaking unless discussed in advance on the mailing

list. FontForge is an extremely complex piece of software, and keeping it in

top form requires this level of care. Please understand that large pull

requests may take a long time to be approved or may be declined, even if well

written, due to limited code review resources.

### How To Contribute, Step by Step

Contribute directly to the codebase using GitHub's Pull Requests.

See [Github Guides](https://guides.github.com/) to learn more, but the basic process is:

- Fork the [FontForge repository](https://github.com/fontforge/fontforge) from GitHub.

- Commit your changes locally using `git`, and push them to your personal fork.

- From the main page of your fork, click on the green “Fork” button in order to submit a Pull

Request.

- Your pull request will be tested via [Travis CI](https://travis-ci.org/) to automatically indicate that your changes do not prevent compilation. FontForge is a big program, so Travis can easily take over 20 minutes to confirm your changes are buildable. Please be patient. More details about using Travis are below.

- If it reports back that there are problems, you can follow the "Details" link to check the log report for your pull request to see what the problem was.

FontForge supports Python `>=` 3.3 and is fully compatible with Python 3 through at least version 3.7.

### Coding Style

Some of these guidelines are not followed in the oldest code in the repository, however we want to use them for all new code since 2012 when FontForge transitioned from a lone-genius project to a collaborative community project.

1. One statement per line, to make semi-automatic processing and reading of diffs much easier.

2. Boolean Variables should use `stdbool.h`'s names `true` and `false`, not an integer ([reference](https://github.com/fontforge/fontforge/issues/724)).

3. `return` statements should be inline with the indentation level they are being put on; don't put them at the left margin as in much of the existing code ([reference](https://github.com/fontforge/fontforge/issues/1208)).

Beyond these guidelines, try to follow the style used in the file being modified.

### People To Ask

Various areas of the codebase have been worked on by different people in recent years, so if you are unfamiliar with the general area you're working in, please feel free to chat with people who have experience in that area:

\* Build System: Debian - Frank Trampe (frank-trampe)

\* Build System: OS X (Application bundle, Homebrew) - Jeremy Tan (jtanx)

\* Build System: Windows - Jeremy Tan (jtanx)

\* Feature: UFO import/export - Frank Trampe (frank-trampe)

\* Feature: Python interface - Skef Iterum (skef)

\* Crashes: Frank Trampe, Adrien Tetar (adrientetar)

### Accessing Travis and Appveyor Build Archives

After each push request `Appveyor` will attempt to build and package

a Windows installer. When that build is successful it can be accessed

by following the `Appveyor` "Details" link and choosing the "Artifacts"

tab.

Note: `Appveyor` builds with the `FF\_PORTABLE` flag, which changes various

initialization and configuration search paths.

The Travis system also attempts to build a Mac OS X application and a

Linux Appimage. When those builds are successful they can be downloaded

from:

https://dl.bintray.com/fontforge/fontforge/

## Translating FontForge

We are trialling the use of Crowdin for handling translations. If you'd like to contribute translations for FontForge, please do so here: https://crowdin.com/project/fontforge

These will be pulled in for the next release of FontForge.

## Building Packages

For information about building FontForge from source, please refer to [INSTALL.md](INSTALL.md)

### Extra Files

If you want to edit CID keyed fonts you need the character set descriptions in [`/contrib/cidmap`](https://github.com/fontforge/fontforge/tree/master/contrib/cidmap)

You might want to pull down some old unicode bitmap fonts.

- [The unifont](http://czyborra.com/unifont/)

- [The FreeFont project](http://www.nongnu.org/freefont/)

- [X fixed](http://www.cl.cam.ac.uk/~mgk25/ucs-fonts.html)

- [Computer Modern Unicode fonts](http://canopus.iacp.dvo.ru/~panov/cm-unicode/) - [Unicode Font Guide for Free/Libre Open Source Operating Systems](http://eyegene.ophthy.med.umich.edu/unicode/fontguide/)

### Building a Debian source package

A Debian source package consists of a source tarball (with specific metadata) and several accompanying files and allows one to build a product in a neutral build environment.

A source package is specific to the distribution (but not the architecture) that it targets.

The most common build target is currently Ubuntu Xenial, a long-term support release with Launchpad build support. A binary package built on and for Xenial (usually) also installs and runs on later versions of Ubuntu.

The first step is to obtain a dist tarball. To generate this from git sources, change into the source directory and run

```bash

mkdir build && cd build

cmake ..

make dist

```

This will generate an archive with a name similar to `fontforge-20190801.tar.xz`. Move/extract this to a working folder of your choice and run `Packaging/debian/setup-metadata.sh`:

```bash

tar axf fontforge-20190801.tar.xz

cd fontforge-20190801

Packaging/debian/setup-metadata.sh

```

This will copy required metadata into the toplevel `debian` folder. It will also prompt to generate `debian/changelog`, if desired. In this state, the Debian source packages may be created by running:

```bash

debuild -S -sa

```

Upon successful completion, building of the source package will leave several files in the parent directory, with names like this:

fontforge-20190801-0ubuntu1~xenial.dsc

fontforge-20190801-0ubuntu1~xenial\_source.build

fontforge-20190801-0ubuntu1~xenial\_source.changes

fontforge-20190801-0ubuntu1~xenial.tar.gz

In order to upload to a Launchpad repository for building, one can then run dput on the `.changes` file with the target repository as the first argument, like this:

dput ppa:fontforge/fontforge fontforge-20190801-0ubuntu1~xenial\_source.changes

This will, upon success, leave a file named something like `fontforge-20190801-0ubuntu1~xenial\_source.ppa.upload` which blocks duplicate uploads.

Upon validation of the uploaded package, Launchpad will build the package for all supported architectures.

One can then copy the binary packages from Xenial to other Ubuntu versions via the Launchpad web interface.

See [here](https://help.launchpad.net/Packaging/PPA) for more information about Launchpad.

One can also build a binary package from the source package locally.

Simply extract the `tar.gz` file generated from `make deb-src` into a new directory, enter the directory, and run `debuild`.

### Building a Red Hat source package (.rpm)

Run the following to get a spec file:

```

Packaging/redhat/generate-spec.sh

```

This will create a `FontForge.spec` file.

In order to build the binary package locally, copy the dist archive (as per Debian instructions) to `~/rpmbuild/SOURCES` and the spec file to `~/rpmbuild/SPECS`. Then run `rpmbuild -ba ~/rpmbuild/SPECS/FontForge.spec`.

For Example:

```

mkdir -p ~/rpmbuild/SOURCES/

cp TARBALL ~/rpmbuild/SOURCES/

rpmbuild -ba --nodeps SPECFILE

```

Upon success, this will leave binary packages in `~/rpmbuild/RPMS` and source packages in `~/rpmbuild/SRPMS`.

You may need to install dependencies, typically packaged for Fedora-derived systems as:

rpm-devel rpm-build git ninja-build cmake gcc g++ python3-devel libjpeg-devel libtiff-devel libpng-devel giflib-devel freetype-devel libxml2-devel libuninameslist-devel libspiro-devel pango-devel cairo-devel gtk3-devel

### Building a Mac OS X app bundle

There is a target to build an app bundle:

```sh

make macbundle

```

This will create a `FontForge.app` in the `osx` subdirectory of your build directory.

This relies on `travis-scripts/ffosxbuild.sh` to make the bundle. It has been tested to work with Homebrew and the GDK backend.