**How to Contribute to Open Source without Coding**

([*Craig Buchek*](http://www.buchek.com/)*to the*[*St. Louis Unix Users Group*](http://www.sluug.org/))

Actually, there are plenty of ways to contribute without coding:

* Submit bug reports
* Suggest new features and options
* Make other comments on how to improve the the quality of the program
* Help write good documentation
* Translate the documentation (and program text) into another language
* Read exisiting documentation, follow the examples, and make corrections
* Correct spelling and grammar mistakes in documentation
* Develop spelling and grammar style conventions for documentors
* Build a glossary of technical terms
* Convert documentation into more useful formats (i.e. DocBook)
* Create templates to write documentation in a WYSIWYG word processor (AbiWord, KWord) and XSLT to transform it into DocBook
* Create diagrams, screen-shots, and graphics for documentation
* Submit graphics (icons, backgrounds) to use in the program
* Help other people learn how to use the program (answer questions on mailing lists or IRC channels)
* Write an email expressing your appreciation for the programs you use
* Send the programmers post cards
* Send the programmers a virtual beer
* Write your legislators about the concerns that Open Source programmers have with recent and upcoming legislation
* Write book reviews and critiques
* Write a book
* Maintain a FAQ or HOWTO document
* Help organize LUG events, including InstallFests, BugFests, and DocFests
* Help write articles for the LUG newsletter
* Help update the LUG web site
* Help maintain a web site for an Open Source project
* Design a better user interface for your favorite program (GLADE and Qt Designer are great for mocking up a new UI)
* Run usability studies
* Create validation or regression test cases
* See how a program handles streams of random data
* Package the application for a particular Linux distro (or other OS)
* Get the program to compile on a new platform
* Create a Linux advocacy web site (probably not so easy to do right)
* Provide training to new Linux users
* Read relevant standards and make sure the program follows them
* Convince people to choose Open Source products when possible
* Write up case studies of successful Open Source implementations
* Send the programmers some money

Here are some suggestions if you want to start coding for an Open Source project:

* Read a lot of code, and learn from that (I’ve never seen a book that stressed this enough, but it is critical, and you’ll read much more than you write, especially with Open Source)
* When reading code, consult include files for info on library functions (Learn how to grep for the function or structure you are looking for)
* Start small, with one-line changes to existing programs (You don’t have to understand too much to do this in many cases)
* Write your own small programs just to learn the language and libraries
* Start off commenting existing code where it needs it
* Write some documentation on the architecture of the program
* Learn how to use all the tools (CVS, diff, patch, libtool, automake…)
* Experiment by making changes to your local copy of the code
* Test your code thoroughly before you submit it
* Adhere to the maintainer’s coding and formatting standards
* Don’t get discouraged when your patches are rejected (they will be!)