Arabeyes Developer Guide

Mohammed Elzubeir, Arabeyes Project <contact (at) arabeyes dot org>

Version 0.2 $Revision: 1.6 $

Revision History

Revision 1.6	2004-04-13	elzubeir
Updated new boostrap example from BiCon. Removed Youcef's name from credits (as per his request). Upped version to 0.2 (guide in effect)		
Revision 1.5	2004-04-06	elzubeir
Modified bootstrapping and source code documentation options.		
Revision 1.4	2004-04-04	elzubeir
Modified Makefile.cvs vs. autogen.sh argument. Removed reference to Akka in favor of BiCon. Modified Alpha release info.		
Revision 1.3	2004-02-17	elzubeir
Noted ArabeyesTodo (wiki) and updated copyright year		
Revision 1.2	2004-02-16	elzubeir
Added link to cvs2cl tool		
Revision 1.1	2004-02-15	elzubeir
XML version of document (convert from SGML)		

This document is meant to serve as a guide for Arabeyes developers and those wanting to join the Arabeyes development team.

Table of Contents

License

Introduction

This document was written in response to the expansion and increase of Arabeyes projects. It became apparent that it was necessary to have a guideline that new and older Arabeyes developers can follow.

There are three main objectives hoped to be achieved by this document:

- To make it easy for new volunteers to learn the ropes of what it takes to start and maintain Open Source projects with Arabeyes.
- To make it easy for users to know what to expect when they intend to use an Arabeyes product.
- To make it easy for new voltunteers who intend to contribute to an Arabeyes project.

Credits

Many thanks to the following individuals who have provided useful feedback when it was needed during the write-up of this document.

- Samy Barha -- for his passionate "tabs vs. spaces" argument.
- Nadim Shaikli -- for his many corrections and recommendations.
- Mohamed Karouri -- for his valuable feedback.

Translations

Currently there no known translations of this document.

Feedback

If you have any questions or suggestion, please do make them known on the 'doc' mailing list here: http://lists.arabeyes.org/mailman/listinfo/doc [http://lists.arabeyes.org/mailman/listinfo/doc]

Getting Started

Arabeyes categorizes development code-projects (ie. non-translation) into two types:

- Internal: projects that are completely written by Arabeyes volunteers.
- External: supplemental patches and fixes to existing applications.

Internal projects are also sometimes termed 'home-brewed' projects since they are completely encompassed within Arabeyes and see/require no external input/direction/approval.

For a comprehensive list of Arabeyes' global TODO's, you may want to visit ArabeyesTodo [http://wiki.arabeyes.org/ArabeyesTodo]. This may give you some ideas as to what you may want to contribute with.

Here are some guidelines on how to proceed -

1. Subscribe to the 'developer' mailing-list on Arabeyes. http://lists.arabeyes.org/mailman/listinfo/developer [http://lists.arabeyes.org/mailman/listinfo/developer].

   - If the project you've select to work and contribute to is an external project, subscribe to the relevant external mailing-lists of the project as well.

2. Post a message to the developer [http://lists.arabeyes.org/mailman/listinfo/developer] list announcing your intentions and plans.

   - If the project you've selected is an external one, make sure that all your posts/correspondence to 'developer' or the external project's mailing-list(s) and/or author(s) get CC'ed to both locations in order to keep all parties involved informed.
   - Guesstimate a time-line and a date by which you expect to complete your tasks (ie. set yourself a deadline and publish it).

3. Research what is required to complete the tasks. This will involve sifting through everything Google [http://www.google.com/] produces as well as various advice you might get from people. In the meantime, start a file with all your notes and findings as well as the various questions (and answers) as they become available.
4. Post your questions/ideas to the 'developer' list.
5. Once satisfied with your research and a working solution is likely, start implementing.
6. Request the creation of a new directory for your project from the Arabeyes CVS Administrator by submitting a bug report on bugs.arabeyes.org [http://bugs.arabeyes.org/].
7. Continue your work and ensure progress (keep that deadline in mind) !!
8. Be very transparent of your progress (make it known); if there are any breaks in your continuation (ie. you go on vacation or something) note it to those working with you without hesitation.
9. It is very important to let others know the progress of the project. Periodically send 'status reports' to the relevant mailing-list. Be sure to link the archived post in a brief update of the project web page as well.

Notes on Deadlines

Deadlines are not set in stone. You set your own time-line as you see fit (there is no need to falsify a 'soon' time-line, realism should be adhered to). If you start realizing that you will be missing your own deadline, push it back/forward as you see fit. However, letting those involved know of these changes is absolutely necessary. If you don't communicate such issues, adverse negative connotations (such as

thinking the project is "dead") might result, thus, it's crucial that open communication remains in place.

Mailing-lists are instrumental to your research and implementation. If you don't use them (on Arabeyes or elsewhere), you are severely decreasing the amount of information you can potentially gather as well as the quality of the end-result.

Lastly, have fun and feel proud, for you are working on something that thousands of people now and in the future will appreciate.

The Source Code

These are some rough guidelines with regard the source code organization and style.

Coding Style Guidelines

Due to the fact that several different programming languages are being used throughout the Arabeyes project it is impossible to set a unifying coding guideline. A good source of information is the GNU Coding Standards [http://www.gnu.org/prep/standards.html].

The most important rule here is consistency. You must be consistent throughout your entire source tree in every convention you do. The way variables, constants, etc. are named must be consistent throughout your entire source tree. The way you indent must be consistent as well (space, tabs, etc.).

File Naming Conventions

All source code files should be in lower-case. This also goes for directory names. In fact, generally all files should be in lower-case. There is one exception to this, which are the files mentioned here.

It is worth noting that there may be some exceptions. For example, Java dictates that you name your files as you have named your classes. It is common programming practice with Java to capitalize your class names. For this reason, it makes sense to capitalize your filenames as well.

The Use of Autotools

It is strongly recommended that you use `autotools` when putting together your project. The use of `autotools` could require a considerable learning curve. Thankfully there are programs that help with this (`autoproject` being one of them). The Autobook [http://sources.redhat.com/autobook/] may also prove to be a valuable resource.

Autotools generate quite a few files in the process. As you probably know by now, files that are generated by other files on CVS are not permitted on CVS. That is, if you have `runme.sh` which creates `x.txt` then you should not have `x.txt` on CVS. It is also not permitted to have files that are generated by standard tools (e.g. `autoconf`) to be present in CVS. For this reason you will need to bootstrap your project. The Bootstrapping [http://sources.redhat.com/autobook/autobook/autobook_43.html] chapter of the Autotools book explains this issue well.

It is recommended that you bootstrap your project depending on the tools used. For instance, if it is a GNOME-based project, it is recommended you use the `autogen.sh` to boostrap. If it is a KDE-based project then it is recommended to use the `Makefile.cvs` to boostrap. However, if your project does not belong to a specific group of toolkits, you should default to the `bootstrap` file as outlined in the Autotools book.

Example 1. bootstrap file

```
#!/bin/sh
#--
# Bootstrap for BiCon
# $Id: developer-guide.en.xml,v 1.6 2004/04/13 08:28:18 elzubeir Exp $
#--

# Run this file to produce a configure script.

DEFAULTARGS="--force --install --autoreconf=auto"

for arg in $DEFAULTARGS $*
do
        case $arg in
                -h | --help)
                        HELP=--help ;;
                -V | --version)
                        VERSION=--version ;;
                -v | --verbose)
                        VERBOSE=--verbose ;;
                -d | --debug)
                        DEBUG=--debug ;;
                -W | --warning | --warnings=yes)
                        WARNINGS=--warnings=all ;;
                --no-warning | --warnings=no)
                        WARNINGS= ;;
                --warning=*)
                        WARNINGS=$arg ;;
                -f | --force | --force=yes | --force-missing)
                        FORCE=--force ;;
                --no-force | --force=no)
                        FORCE=--no-force ;;
                -i | --install | --install=yes | -a | --add-missing)
                        INSTALL=--install ;;
                --no-install | --install=no)
                        INSTALL= ;;
                -s | --symlink | --symlink=yes | --no-copy | --copy=no)
                        SYMLINK=--symlink ;;
                --no-symlink | --symlink=no | --copy | --copy=yes)
                        SYMLINK= ;;
                -m | --make | --make=yes)
                        MAKE=--make ;;
                --no-make | --make=no)
                        MAKE= ;;
                -n | --dry-run)
                        DRYRUN=echo ;;
                --autoreconf=auto)
                        AUTORECONF=auto ;;
                --autoreconf | --autoreconf=yes)
                        AUTORECONF=yes ;;
                --no-autoreconf | --autoreconf=no)
                        AUTORECONF= ;;
                *)
                        echo Ignoring unknown parameter $arg
        esac
done

test -z "$SYMLINK" && COPY=--copy
test -n "$INSTALL" && ADDMISSING=--add-missing

# use autoreconf if possible, just check for version 2+
if test "$AUTORECONF" = auto; then
        case `autoreconf --version 2>/dev/null` in
                *"autoreconf (GNU Autoconf) 2."* )
```

```
                        echo Usable autoreconf found, running
                ;;
                *)
                        AUTORECONF=
                ;;
        esac
fi

if test -n "$AUTORECONF";  then
        $DRYRUN autoreconf $HELP $VERSION $VERBOSE $DEBUG $FORCE $INSTALL $SYMLINK
        exit $?
fi

# add files 'config.guess', 'config.sub', 'ltconfig', 'ltmain.sh'
$DRYRUN libtoolize $HELP $VERSION --automake $COPY $DEBUG $FORCE || exit 1

# generate 'aclocal.m4'
if test -f configure.ac -o configure.in; then
        $DRYRUN aclocal $HELP $VERSION $VERBOSE $FORCE || exit 1
fi

# generate 'config.h.in'
if test -f configure.ac -o configure.in; then
        $DRYRUN autoheader $HELP $VERSION $VERBOSE $DEBUG $FORCE $WARNINGS || exit
fi

#  generate Makefile.in's from Makefile.am's
if test -f Makefile.am; then
        $DRYRUN automake $HELP $VERSION $VERBOSE $ADDMISSING $COPY $FORCE $WARNING
fi

# generate configure from configure.ac
if test -f configure.ac -o -f configure.in; then
        $DRYRUN autoconf $HELP $VERSION $VERBOSE $DEBUG $FORCE $WARNINGS || exit 1
fi

if test -n "$MAKE"; then
        if test -f ./configure; then
                $DRYRUN ./configure $HELP $VERSION || exit 1
        fi
        if test -f Makefile; then
                $DRYRUN make || exit 1
        fi
fi
```

Feel free to use the above in your program, with whatever modifications you may need to make. Different people have different tastes in how they want to boostrap their project. As long as you do have a sensible mechanism to boostrap, you are good to go, but whatever you do, you cannot leave auto-generated files from autotools in CVS.

Project Documentation

RCS Header

There are two types of RCS headers to use, depending on the relative size of the file. If it is a large file then we use the full RCS header. All source code also must use the full RCS header (*.c, *.h, *.cc, *.cpp, *.java, etc.).

Example 2. Full RCS Header

```
/*************************************************************************
 * $Id: developer-guide.en.xml,v 1.6 2004/04/13 08:28:18 elzubeir Exp $
 *
 * ------------
 * Description:
 * ------------
 *  Copyright (c) 2004, Arabeyes, YOUR NAME
 *
 *  HERE ENTER A DESCRIPTION OF THE FILE AND ITS PURPOSE
 *
 * -----------------
 * Revision Details:    (Updated by Revision Control System)
 * -----------------
 *  $Date: 2004/04/13 08:28:18 $
 *  $Author: elzubeir $
 *  $Revision: 1.6 $
 *  $Source: /home/arabeyes/cvs/doc/guide/developer/developer-guide.en.xml,v $
 *
 *************************************************************************/
```

However, if it is a small file, like a `TODO` file or something similar, then we use a small RCS header. Remember that when we say small file, it is not the size in byes only. For example, an obvious file that does not require explanation does not warrant a full RCS header. Use common sense here.

Example 3. Small RCS Header

```
#--
# filename -- short description (one line)
# $Id: developer-guide.en.xml,v 1.6 2004/04/13 08:28:18 elzubeir Exp $
#--
```

Note that all RCS keywords are enclosed between $ signs. It is automatically expanded once you cvs commit your files.

Source Code Documentation

All source code must be commented. This allows you to generate documentation for your program without having to write extra documentation for the API. The commenting style will depend on the toolset(s) and language(s) used by your project. For example, Java has javadoc which you can use. The same applies to GNOME, KDE and the likes. In the case of not belonging to a group of projects that applies its own source code documentation style, you must default to Doxygen.

Making Doxygen style comments is not hard or complicated. It only requires a negligible learning curve and you should be up and running in less than 5 minutes. The point of this whole exercise is to ensure a certain level of quality of source code. Documenting your code not only allows others to be able to follow your code, it allows you to re-evaluate your logic and helps you in maintaining your code.

You will need to create a `Doxyfile` which holds the necessary configuration options to generate the documentation. There are also GUI tools that help you in creating this configuration file.

The following is an example from Duali (C++):

Example 4. Doxygen configuration file

```
#--
# Duali Doxygen Configuration File
# $Id: developer-guide.en.xml,v 1.6 2004/04/13 08:28:18 elzubeir Exp $
#--

PROJECT_NAME           = Duali
PROJECT_NUMBER         = 0.2
OUTPUT_DIRECTORY       = ../doc
OUTPUT_LANGUAGE        = English
EXTRACT_ALL            = YES
EXTRACT_PRIVATE        = YES
EXTRACT_STATIC         = YES
EXTRACT_LOCAL_CLASSES  = YES
HIDE_UNDOC_MEMBERS     = NO
HIDE_UNDOC_CLASSES     = NO
HIDE_FRIEND_COMPOUNDS  = NO
HIDE_IN_BODY_DOCS      = NO
BRIEF_MEMBER_DESC      = YES
REPEAT_BRIEF           = YES
ALWAYS_DETAILED_SEC    = NO
INLINE_INHERITED_MEMB  = NO
FULL_PATH_NAMES        = NO
STRIP_FROM_PATH        =
INTERNAL_DOCS          = YES
CASE_SENSE_NAMES       = YES
SHORT_NAMES            = NO
HIDE_SCOPE_NAMES       = NO
VERBATIM_HEADERS       = YES
SHOW_INCLUDE_FILES     = YES
JAVADOC_AUTOBRIEF      = YES
MULTILINE_CPP_IS_BRIEF = NO
DETAILS_AT_TOP         = YES
INHERIT_DOCS           = YES
INLINE_INFO            = YES
SORT_MEMBER_DOCS       = YES
DISTRIBUTE_GROUP_DOC   = NO
TAB_SIZE               = 4
GENERATE_TODOLIST      = YES
GENERATE_TESTLIST      = YES
GENERATE_BUGLIST       = YES
GENERATE_DEPRECATEDLIST= YES
ALIASES                =
ENABLED_SECTIONS       =
MAX_INITIALIZER_LINES  = 30
OPTIMIZE_OUTPUT_FOR_C  = NO
OPTIMIZE_OUTPUT_JAVA   = NO
SHOW_USED_FILES        = YES
#---------------------------------------------------------------------------
# configuration options related to warning and progress messages
#---------------------------------------------------------------------------
QUIET                  = NO
WARNINGS               = YES
WARN_IF_UNDOCUMENTED   = YES
```

```
WARN_IF_DOC_ERROR      = YES
WARN_FORMAT            = "$file:$line: $text"
WARN_LOGFILE           =
#---------------------------------------------------------------------------
# configuration options related to the input files
#---------------------------------------------------------------------------
INPUT                  = /home/elzubeir/cvs/arabeyes/projects/duali/src/
FILE_PATTERNS          =
RECURSIVE              = NO
EXCLUDE                =
EXCLUDE_SYMLINKS       = NO
EXCLUDE_PATTERNS       =
EXAMPLE_PATH           =
EXAMPLE_PATTERNS       =
EXAMPLE_RECURSIVE      = NO
IMAGE_PATH             =
INPUT_FILTER           =
FILTER_SOURCE_FILES    = NO
#---------------------------------------------------------------------------
# configuration options related to source browsing
#---------------------------------------------------------------------------
SOURCE_BROWSER         = YES
INLINE_SOURCES         = YES
STRIP_CODE_COMMENTS    = YES
REFERENCED_BY_RELATION = YES
REFERENCES_RELATION    = YES
#---------------------------------------------------------------------------
# configuration options related to the alphabetical class index
#---------------------------------------------------------------------------
ALPHABETICAL_INDEX     = NO
COLS_IN_ALPHA_INDEX    = 5
IGNORE_PREFIX          =
#---------------------------------------------------------------------------
# configuration options related to the HTML output
#---------------------------------------------------------------------------
GENERATE_HTML          = YES
HTML_OUTPUT            = html
HTML_FILE_EXTENSION    = .html
HTML_HEADER            =
HTML_FOOTER            =
HTML_STYLESHEET        =
HTML_ALIGN_MEMBERS     = YES
GENERATE_HTMLHELP      = NO
CHM_FILE               =
HHC_LOCATION           =
GENERATE_CHI           = NO
BINARY_TOC             = NO
TOC_EXPAND             = YES
DISABLE_INDEX          = NO
ENUM_VALUES_PER_LINE   = 4
GENERATE_TREEVIEW      = YES
TREEVIEW_WIDTH         = 250
#---------------------------------------------------------------------------
# configuration options related to the LaTeX output
#---------------------------------------------------------------------------
GENERATE_LATEX         = NO
LATEX_OUTPUT           = latex
LATEX_CMD_NAME         = latex
MAKEINDEX_CMD_NAME     = makeindex
COMPACT_LATEX          = NO
PAPER_TYPE             = a4wide
EXTRA_PACKAGES         =
LATEX_HEADER           =
PDF_HYPERLINKS         = NO
```

```
USE_PDFLATEX           = NO
LATEX_BATCHMODE        = NO
#---------------------------------------------------------------------------
# configuration options related to the RTF output
#---------------------------------------------------------------------------
GENERATE_RTF           = NO
RTF_OUTPUT             = rtf
COMPACT_RTF            = NO
RTF_HYPERLINKS         = NO
RTF_STYLESHEET_FILE    =
RTF_EXTENSIONS_FILE    =
#---------------------------------------------------------------------------
# configuration options related to the man page output
#---------------------------------------------------------------------------
GENERATE_MAN           = NO
MAN_OUTPUT             = man
MAN_EXTENSION          = .3
MAN_LINKS              = NO
#---------------------------------------------------------------------------
# configuration options related to the XML output
#---------------------------------------------------------------------------
GENERATE_XML           = NO
XML_SCHEMA             =
XML_DTD                =
#---------------------------------------------------------------------------
# configuration options for the AutoGen Definitions output
#---------------------------------------------------------------------------
GENERATE_AUTOGEN_DEF   = NO
#---------------------------------------------------------------------------
# configuration options related to the Perl module output
#---------------------------------------------------------------------------
GENERATE_PERLMOD       = NO
PERLMOD_LATEX          = NO
PERLMOD_PRETTY         = YES
PERLMOD_MAKEVAR_PREFIX =
#---------------------------------------------------------------------------
# Configuration options related to the preprocessor
#---------------------------------------------------------------------------
ENABLE_PREPROCESSING   = YES
MACRO_EXPANSION        = NO
EXPAND_ONLY_PREDEF     = NO
SEARCH_INCLUDES        = YES
INCLUDE_PATH           =
INCLUDE_FILE_PATTERNS  =
PREDEFINED             =
EXPAND_AS_DEFINED      =
SKIP_FUNCTION_MACROS   = YES
#---------------------------------------------------------------------------
# Configuration::addtions related to external references
#---------------------------------------------------------------------------
TAGFILES               =
GENERATE_TAGFILE       =
ALLEXTERNALS           = NO
EXTERNAL_GROUPS        = YES
PERL_PATH              = /usr/bin/perl
#---------------------------------------------------------------------------
# Configuration options related to the dot tool
#---------------------------------------------------------------------------
CLASS_DIAGRAMS         = YES
HIDE_UNDOC_RELATIONS   = YES
HAVE_DOT               = NO
CLASS_GRAPH            = YES
COLLABORATION_GRAPH    = YES
TEMPLATE_RELATIONS     = YES
```

```
INCLUDE_GRAPH          = YES
INCLUDED_BY_GRAPH      = YES
GRAPHICAL_HIERARCHY    = YES
DOT_IMAGE_FORMAT       = png
DOT_PATH               =
DOTFILE_DIRS           =
MAX_DOT_GRAPH_WIDTH    = 1024
MAX_DOT_GRAPH_HEIGHT   = 1024
GENERATE_LEGEND        = YES
DOT_CLEANUP            = YES
#---------------------------------------------------------------------------
# Configuration::addtions related to the search engine
#---------------------------------------------------------------------------
SEARCHENGINE           = NO
CGI_NAME               = search.cgi
CGI_URL                =
DOC_URL                =
DOC_ABSPATH            =
BIN_ABSPATH            = /usr/local/bin/
EXT_DOC_PATHS          =
```

The above is simply an example and is not meant to be followed rigidly. However, it is expected you do indeed provide proper documentation throughout your code to comply with Doxygen commenting formats. For more information on that, please do consult your friendly Doxygen manual [http://www.stack.nl/~dimitri/doxygen/manual.html] (or look at Duali as a possible example).

Manpages

It is extremely important that manpages are created for any project. In fact, no releases are to be made without manpages to accompany your program. It is simply unacceptable to not have a manpage for your program. However, you may feel free to also include `info(1)` pages.

Manpages are relatively simple to make. Here is an example of a manpage:

Example 5. A manpage

```
.\"                                      Hey, EMACS: -*- nroff -*-
.\" First parameter, NAME, should be all caps
.\" Second parameter, SECTION, should be 1-8, maybe w/ subsection
.\" other parameters are allowed: see man(7), man(1)
.TH DUALI 1 "Aug 16, 2003"
.\" Please adjust this date whenever revising the manpage.
.\"
.\" Some roff macros, for reference:
.\" .nh        disable hyphenation
.\" .hy        enable hyphenation
.\" .ad l      left justify
.\" .ad b      justify to both left and right margins
.\" .nf        disable filling
.\" .fi        enable filling
.\" .br        insert line break
.\" .sp <n>    insert n+1 empty lines
.\" for manpage-specific macros, see man(7)
.SH NAME
duali \- The Arabic Spell Checker.
.SH SYNOPSIS
```

```
.B duali
.RI \-c\ filename\ [OPTIONS] ...
.SH DESCRIPTION
.PP
.\" TeX users may be more comfortable with the \fB<whatever>\fP and
.\" \fI<whatever>\fP escape sequences to invode bold face and italics,
.\" respectively.
\fBduali\fP is an open source cross-platform Arabic spell checker.

.SH OPTIONS
.TP
.B \-h, \-\-help
Show summary of options.
.TP
.B \-V, \-\-version
Show version of program.
.TP
.B \-c, \-\-check= filename
File to spellcheck (required)
.TP
.B \-C, \-\-charset
Select file character encoding (default utf-8). The options are between utf-8
and cp1256. It is recommended that you only use utf-8.
.TP
.B \-n, \-\-normalize
Switch normalize mode off (default on). The normalize mode strips all diacritic
marks from the spell checked text (recommended).
.TP
.B \-p, \-\-path
Path to dictionary database (this option should probably be set in your
configuration file).
.TP
.B \-v, \-\-verbose
Verbose out (for debugging)

.SH AUTHOR
This manual page was originally written by Mohammed Sameer <uniball (at) gmx dot n
and later modified and adapted by Mohammed Elzubeir <elzubeir (at) arabeyes dot or
```

To learn more about creating manpages, you can always go through the Linux Man Page HOWTO [http://www.tldp.org/HOWTO/Man-Page/index.html].

TODO Lists

No program is ever complete. There is always room for improvement and enhancements. That is why we have TODO lists. Arabeyes projects employ two types of TODO lists: Web-based and text-based. It is the responsibility of the project maintainer to synchronize between the two lists.

The web-based TODO list manager was deployed in order to help project maintainers make it easier for new volunteers to easily find what they can help with. It uses a simple to use web interface and should be rather intuitive.

The text-based TODO's are what you will be placing in CVS. This usually sits in the root of your CVS module. Although it is not mandatory, it is strongly urged that all projects do maintain an up-to-date TO-DO list.

The format should also be consistent. The format should be as the following example demonstrates:

Example 6. TODO List

```
#--
# XXX TODO List
# $Id: developer-guide.en.xml,v 1.6 2004/04/13 08:28:18 elzubeir Exp $
#--

+ do x1 and x2 again
+ yet another item.. see, it's very simple
+ do x and do y while maintaining that z remains cool
  (*) DONE - Mar. 23, 2001
```

Be sure to keep the "DONE" items at the bottom while the active items on top.

ChangeLogs

It is always nice to be able to find out what has changed from release to another. If you haven't been updating your ChangeLog as you go along it could be a rather tedious process that you would want to completely put off. However, no release can be made without a `ChangeLog` file.

With that being said, `cvs2cl` is to the rescue (http://www.red-bean.com/cvs2cl! Remember how you have been working very hard at making sure your CVS commit logs reflect the work you have been doing? This is one area where it pays off. `cvs2cl` will generate a ChangeLog based on those CVS commit logs. You can then go through its output and modify it accordingly.

Example 7. ChangeLog

```
#--
# BiCon Change Log
# $Id: developer-guide.en.xml,v 1.6 2004/04/13 08:28:18 elzubeir Exp $
#--

2004-04-01 08:33  behdad

        * README, bin/bicon.in, doc/man/Makefile.am, doc/man/bicon.1,
          doc/man/bicon.bin.1: Wow, now bicon(1) works for X terminals too!

2004-04-01 06:44  behdad

        * bicon/.cvsignore: One more.  Sorry for the noise ;).

2004-04-01 06:44  behdad

        * .cvsignore, bconsole/.cvsignore, bjoining/.cvsignore,
          font/.cvsignore, keymap/.cvsignore: .cvsignore added for each
          directory that needs it.  Happy "cvs up"ing :).

2004-04-01 06:43  behdad

        * CVSROOT/cvsignore: Added, with typicall autotools based common
          files to ignore.
```

Patches

There are times when you are not the maintainer of a project or do not have CVS access but would like to contribute with code (whether it is enhancements or bug fixes). This is particularily true to new members who need to prove themselves to a given project's maintainer so he/she would allow them to directly commit to their source tree. It is also possible that the contributor simply wants to make one (or more) contributions but not get directly involved in the project. In either case, a patch is the answer.

Other than the fact that it is easier to deal with patches when wanting to integrate it back into the main code, it makes it much easier for the developer and/or maintainer of a project to review the changes proposed.

To create a patch for changes on a single file, you can simply do:

```
$ diff -uB file.c.orig file.c > my.patch
```

Make sure to have the original file first and then the modified one second. This would avoid the headache of reverse patches that often result when patches are created in a hurry.

In the case where the modifications span across multiple files and directories you can use the **-r** switch.

```
$ diff -uBr dir.orig/ dir/ > my.patch
```

In some cases, there may be newly created files not found in the original distribution of a given project. In this case, we use the **-N** switch.

One thing to note when creating patches recursively, espcially with the use of the **-N** switch is to ensure the original directory is a virgin one. That is, without bootstrapping the project or autoconfiguring it. That is because bootstrapping creates many files that will automatically be added to the patch, causing the maintainer much unnecessary pain!

Once you have created the patch you will need to submit it via the bug tracking system used by Arabeyes, bugzilla [http://bugs.arabeyes.org/].

Project Releases

Project releases must be coordinated with all parties actively involved in the project. There are several other things involved before making a release.

Pre-requisites

Miscellaneous Required Files

There are some files that must be there while others are optional. The mandatory files are:

- README -- miscellaneous notes.
- INSTALL -- installation instructions.
- AUTHORS -- list of author. It should simply be the author name followed by the email address, line

by line.

- COPYING -- this file should contain the license that is being used for the project, be it GPL, BSD or any other Open Source license.
- ChangeLog -- this file has been discussed earlier in theChangeLog

There are always people who have helped with the project in one way or the other but are not listed in the `AUTHORS` file. This is where you can mention them in a file named `THANKS`. Always give credit where it is due. That is an absolute must.

Be Sure It works

This may seem obvious, but mistakes do happen. It is not uncommon to have several directories with different versions of the software you are working on. Once you have decided what files you will include in your release, make a copy of them in a new directory and try compiling it then running it. Ensure that you have all the right and necessary files there.

It may be best to use the cvs command **cvs export** in order to create a copy of the source tree without the CVS administrative files. You may also want to automate this process as much as possible. However, it is entirely up to you how you handle this.

Tarballs

All tarballs should be named as follows: `pkgname-verion.tar.gz` (or .tar.bz2/.tgz/etc.). They should also untar into a directory by the name of `pkgname-version/`. It should never untar into the current working directory or simple `pkgname/`.

Version System

In order to make it easy for a user to know what to expect from a particular project's version, we will use certain conventions in our version numberings.

The version scheme should be <major-revision>.<minor-revision>.< bugfix-revision>. The major revision reflects major changes in the program. The minor revision reflects the general status of the code (beta or stable). The bug-fix revision is optional. They constitute changes that only involve bug fixes or very minor changes. The only exception to the above rule are alpha releases.

Alpha releases

Alpha releases are what one can call, 'bleeding-edge' releases. Alpha releases often contain new code with features not thoroughly tested yet. Alpha releases should only have <major>.<minor> version numbers attached to them. In the case of a new project which does not have any release yet, you will use 0.0 as the version number. For example, `pkgname-0.0.20030428`.

Beta releases

Beta releases are one step better than alpha releases. Generally beta releases should be released after some feedback was received post an earlier alpha release. Beta releases are denoted by odd version numbers. For example, `pkgname-1.3`.

Stable releases

Stable releases should only be made when feedback and tests have been performed not only by the developers and active testers, but by end-users who are not interested in the development of the project

(but only the use of it). This is particularly important to observe, in order to maintain a high standard of expectations for all releases.

Stable releases should simply include the package name followed by an even version number. For example, `pkgname-1.4`.

Release Candidates

Before a major release, sometimes it is preferable to do 'last-minute' testing. Generally this should only be used for large projects. However, a maintainer may elect to release several quick and small releases without wanting to increment the version number. This is particularly true when the changes are quick and small bug-fixes.

Release candidates should have the package name followed by 'RCx' where x is a number to be incremented as necessary (starting from 1). For example, `pkgname-1.5RC2`, is version 1.5 release candidate 2.

It should be noted however, that this option should only be used for very large projects (like Arabbix for example). Smaller projects should not have to deal with release candidates.

CVS Tags

Whenever you are about to make a release, you must tag the CVS. This makes it possible for you to go back to a specific version from CVS without having to remember every file's revision number.

Tag names should be as follows `PKGNAME_X_X`. Also note that the tag should be in all capital letters. If it is an alpha release then simply name it `PKGNAME_XXXXXXXX`.

To tag your project, simply issue the 'cvs tag' command. For example:

```
$ cvs tag -R QAMOOSE_2_0
```

Release Checklist

Once you have prepared your tarball and any other distro-specific package (be it DEB or RPM or any other package), it is time to release your program to the world. You need to let the world know!

1. Upload the files on ftp://upload.sourceforge.net. If you are a project maintainer and do not have a sf.net account, create one. If you do have one but are not listed as one of the developers on the Arabeyes project then request to be added by emailing 'contact (at) arabeyes dot org'.

   You should simply follow the instructions provided by SourceForge [http://sourceforge.net/] on how to add a new release.

2. Announce on Arabeyes.org

   - Post an announcement on the Arabeyes announce [http://lists.arabeyes.org/mailman/listinfo/announce] list. Be sure to include important information such as: version number, what's new and the changelog.

   - Post a short announcement on the project page (if you are the project maintainer you should have access to do that). Keep it short and to the point. Link to the archived announce message you posted earlier on the 'announce' list.

- Post an annoucement on the Arabeyes general [http://lists.arabeyes.org/mailman/listinfo/general] list. Unfortunately not everyone is on the 'announce' list and there are more people on the 'general' list. For this reason it is a good idea to repost the announcement on the 'general' list. Please do NOT cross-post (post to announce and Cc general). This is deeply frowned upon and will cause the list manager to yell and scream at you ;)

3. Announce on Freshmeat.net [http://freshmeat.net/]. Arabeyes projects are listed under the user 'arabeyes'. If your project is new, please do ask one of the Arabeyes managers to add your project and do the first announcement for you.

   If your project is already listed then simply add a new release. The projects are set that anyone with a freshmeat.net account can update. We use the honor system here, until something goes wrong ;)

4. Announce it anywhere else you like (e.g. linux4arab.com [http://www.linux4arab.com/], arabiclinux.org [http://www.arabiclinux.org/], arabiclinux.com [http://www.arabiclinux.com/], linux-egypt.org [http://www.linux-egypt.org/], etc.). This depends on how important you think your release is (or how much time you have to spend on it!) The general rule is, the more people learn about it, the more likely you will get attention.

5. Ask the Arabeyes bugzilla maintainer to add the new release version to your project in http://bugs.arabeyes.org/. This can be done by simply submitting a bug report on bugs.arabeyes.org [http://bugs.arabeyes.org/].

Now the world knows about your project and your new release. Wait for the feedback! Very often, people will not offer feedback when you are asking for it before the release. Once you make your release you will notice that people will start giving you feedback. That is why, it is always good to release early and release often.