Arabeyes CVS HOWTO

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

$Revision: 1.4 $

Revision History

Revision 1.4	2004-04-14	elzubeir
Added info on how to generate encrypted password.		
Revision 1.3	2004-04-13	elzubeir
Restructured document outline.		
Revision 1.2	2004-02-17	elzubeir
Updated information on account requests.		
Revision 1.1	2004-02-17	elzubeir
XML'ized SGML version of document.		

This document is meant to show you how to access and use the Arabeyes CVS repository.

Table of Contents

Introduction

CVS stands for Concurrent Versions System. It is a version control system that has been developed in the public domain by many people beginning in 1986. Using CVS, you can record the history of your source files and/or documents. It does not check your work for you, but it is a good safeguard against disasters, should they happen.

This HOWTO is not going to teach you everything about CVS, but it is designed to be a quick guide to using the CVS Repository. For more information on using CVS you can check:

- CVSHome - http://www.cvshome.org/
- Intro to CVS - http://linux.oreillynet.com/pub/a/linux/2002/01/03/cvs_intro.html [http://linux.oreillynet.com/pub/a/linux/2002/01/03/cvs_intro.html]

There are many free cvs clients you can download. If you are a Linux/Unix user, then you probably have it as part of your standard install. If you are a Windows user, you have two options:

- Cygwin bash shell - http://sources.redhat.com/cygwin/ [http://sources.redhat.com/cygwin/]

  Cygwin is a UNIX environment for Windows, which comes with a command-line cvs client.

- WinCVS - http://www.wincvs.org/

  WinCVS is a graphical cvs client for MS-Windows

It is strongly recommended that you use the linux/unix cvs client. However, if you must use Windows, Cygwin bash shell is what we recommend. This HOWTO assumes that you either know the equivalent of commands on the graphical front-end, or use the command-line interface.

Access CVS

There are two types of CVS access: Anonymous (read-only access; open to all), and developers (read/write access;restricted).

To access the CVS server (known as 'pserver'), you will need to have an account (username and password). For the purpose of this document, we will assume your username is *myaccount* and password is *rocks*.

Gain Anonymous Access

Anonymous access is the same as the account based access, except that you would not be able to commit files to the CVS repository. The username is *anoncvs* and password is *anoncvs*.

```
$ cvs -d:pserver:anoncvs@cvs.arabeyes.org:/home/arabeyes/cvs login
$ cvs -d:pserver:anoncvs@cvs.arabeyes.org:/home/arabeyes/cvs co module
```

In the above example, be sure to replace the *module* with the approrpriate module name from the CVS repository.

Gain Commit Access

Request Commit Accounts

CVS write access is not lightly given. You may want CVS access if you are:

- Starting a new project
- Contributing to an existing project

If you are requesting an account to start a new project, then your account will be approved once the Core Management Team has apporoved your project. However, if you are requesting an account to contribute to an existing project, then the coordinator of that particular project will have to approve your account creation first.

In order to request CVS write access on Arabeyes' CVS repository, you must file a bug-report via bugs.arabeyes.org [http://bugs.arabeyes.org/]. It is also expected that you have read this document and know exactly why you are needing CVS access.

If at any point you wish to change your CVS password, you will need to email the CVS administrator with an *encrypted* version of your password. This can be done by taking the output of one of the following methods:

```
$ perl -e 'print crypt "PASSWORD", "XY"'
```

Or

```
$ python -c 'import crypt; print crypt.crypt("PASSWORD", "XY")'
```

Where PASSWORD is your password and XY are two random alphanumeric characters of your choice.

Log In

The first thing to do is to login! Here are some vital pieces of information you are going to need set your CVSROOT, like:

```
$ export CVSROOT=:pserver:myaccount@cvs.arabeyes.org:/home/arabeyes/cvs
```

If you are using WinCVS then you will need to break it down to:

- Authentication type - pserver
- Username - myaccount
- Host name - cvs.arabeyes.org
- CVSROOT - /home/arabeyes/cvs

Once you have your environment set, you can then login as follows:

```
$ cvs login
```

Enter your password when it prompts you for one. You are in!

Use CVS

Check Files Out (Download)

The word 'checkout'[1] in CVS refers to downloading the files to your local machine. It is analogous to you checking out a book from the library, however you are not only expected to bring it back, but update it too!

[1] 'checkout' and 'co' are the same thing in the command-line - you can use both interchangeably

```
$ cvs checkout projects
```

The above command will create a directory structure on your computer, which will have everything that is under projects on the CVS repository. So, let us say we don't want everything under projects. Let us assume that you want the Akka project source and everything for it.

```
$ cvs checkout projects/akka
```

This should download everything under `akka/`.

Commit Changes (Upload)

When you 'commit' a file, you are essentially uploading it to the repository. So, now that you have downloaded the files and have made some changes, you would like to put it back on the repository.

For the sake of this example, we will say you are modifying the file 'myfile.cc'. Here is how to commit it:

```
$ cvs commit myfile.cc
```

If the file does not already exist, and you are creating a new one then you need to 'add' it first. Here is how:

```
$ cvs add myfile.cc
$ cvs commit myfile.cc
```

The first command creates an entry for the file in the repository, and the second one actually makes the commit to the repository (making it available). If the file already exists and you are making changes to it, you can ignore the 'cvs add' command. For more help do 'man cvs' from your linux/unix prompt.

Once you execute the commit command, a text editor will be launched so you can enter a brief log of what you have done. This is especially useful for others who are working on the same project - as it gives them an insight of what you have changed without having to look through the file(s). The editor is usually vi, on most Linux/Unix systems.However, that is not necessarily the case. It is whatever you have the environment $EDITOR set to. To find out what it is set to in your environment, do:

```
$ env | grep EDITOR
```

In order to get into *insert* mode under VI, press **i**. Then you may continue to type in your brief log. Once you are done, press **ESC** (which will get you out of the insert mode), and enter **:wq** then hit return. This will write (to save) and quit. Your commit will then be executed.

Commit Multiple Files

When you have made changes that span across multiple files it is best to commit them in one batch, as

opposed to doing so one at a time. That is for 2 main reasons:

- Minimizes on the amount of cvs notification mail that goes out to everyone on the 'cvs' list, whenever a commit is made.
- Most of the time the multiple files changed involve the same issue. For example, if the two files `test.cc` and `test.h` were changed, it is unnecessary to commit them individually.

There are two ways of committing multiple files.

```
$ cvs commit file1 file2 file3
```

Or you can simply let it recursively do it:

```
$ cvs commit
```

The first gives you more flexibility in what the files are exactly, whereas the latter simply commits any file that has changed in the current directory and all subdirectories.

Create New Directories

New directories are either created as part of an existing project or to create a new module for a newly imported project. Before creating any new directory, please always consult the CVS administrator. This is particularily important to avoid redundancy and maintain an ordered organization.

This does not apply to those who are module owners. For instance, if the author of X decided to add `X/yz/` to his `X/` directory, it does not require any approval from the CVS administrator.

Although this does not happen very often, it is worth noting here, especially for those who own a project and would like to add a new directory with some content in it already.

When that is the case, use **cvs import**. For example, if projX in `projects/projX` is about to add a new directory called `games` with 10 files in it, do this:

```
$ cvs import projects/projX/games arabeyes start
```

For more information type **cvs import --help**' or consult the CVS Manual.

Write Commit Logs

It goes without saying that it is very important to enter a useful and meaningful comment on commits.

- Translation (PO's):

  When committing `.po` file(s) always make sure the comment includes the output of:

```
$ msgfmt --statistics file.po 2
```

This is to avoid commit logs such as "translated a little", which mean very little if anything at all.

- Development Source Code:

  Comments on source code commits may be a little easier (as the types of changes will vary). Always make sure you keep track of the changes you make *AS* you make them, so you remember what they are before committing.

  An example of a bad log is: "I made a few fixes". What fixes? If you can't remember what you did anymore, then do a diff of the changes, go through it and note the changes you made.

- Documents:

  Generally documentation comments are inevitably going to be vague. If the document is small, that's not a problem. If it is not, point to the location you made the changes to. For example: "Chapter 3, Section 4 - Re-wrote the second paragraph for clarity" is a good comment.

Commit The Right Files

There is a very simple rule to follow on this one. Any file that is generated by a program from another file that resides on CVS should not be put on CVS. That is unnecessary redundancy.

Compressed files (gzip/bzip2/Z/etc) should not be put on CVS in their compressed/tarred forms.

Generally the only binary type of files acceptable on CVS are graphics images and audio files. Every other file type is expected to be text.

Update Your Working Copy

Since there are others who are likely to have made changes during the course of the project, your local working copy of the cvs contents may not always be up to date. In order for you to keep your copy up-to-date you can:

```
$ cvs update
```

Note that this can only be done while you are inside the directory you are intending to update.

2
'msgfmt' is a utility that comes with the gettext package.