Introduction to CVS

14 October 2010

Prepared by Mostafa Shad

CVS: Concurrent Versions System is a program which provides a way to record the revision history of any document and allows for concurrent editing file.

Terminology:

1. **Repository:** Central location where CVS server stores its files. In our server, it is located at /cvsroot.
2. **Sandbox:** Local copy of a directory tree checked out from a repository.
3. **Module:** Directory tree or trees that pertain to a single project.

Installation

sudo apt-get install cvs

Environment Variables:

1. **CVSROOT:** contains repository root directory, use following command to set this variable:

export CVSROOT=/cvsroot

2. **CVS_RSH:** remote shell command.

export CVS_RSH=ssh

Some Basic Commands:

General form of CVS command is as follow:

cvs {<global option>} <command> {<command option>} name …

Common global options are:

-d	<dir>	Specify repository dir
-m	<msg>	Use <msg> as the log message
-D	<date>	Use the latest version as of <date>

1. **Creating a Repository**

   mkdir /cvsroot

   cvs –d /cvsroot init

2. **Importing a Module**

   mkdir module_name

   cd module_name

   cvs import –m 'improting new project' module_name vendor_tag release_tag

3. **Checking out a Module:** this command creates a sandbox of the module in working directory

   cd local_dir

   cvs co module_name

   These command creates a directory (module_name) at current working directory containing module files and a CVS directory with following files:

   - Entries: contains a line for each file or directory in the current directory
   - Repository: contains the location in the repository of this module
   - Root: contains the location of the root of the CVS repository

4. **Adding a File:** adding a file to a module has 2 steps. First use '*add*' command to locally add the file to the module and then use '*commit'* command to update the CVS.

   cd module_name

   cat >main.c << 'EOF'

   //sample code

   EOF

   cvs add main.c

   cvs commit –m 'initiating main code' main.c

5. **Checking what is changed in edited file:** if the main.c changes as follow:

   //sample code

   int main()

   {

   return 0;

   }

   Before committing the changes, you can see what exactly was changed by following command:

   cvs diff main.c

6. **Updating:** if someone changes the repository source files, you can also update your sandbox and get the latest version of files. To see the changes, use following commands:

   cvs status

   cvs update

7. **Merge:** consider user1 changes main.c and commit the changes while user2 is still using previous edition of the files and making changes on earlier revision. If user2 tries to commit changes, following error will be generated:

   cvs commit –m 'user2 changes' main.c

cvs commit: Up-to-date check failed for 'main.c'

cvs [commit aborted]: correct above error first

User2 must first update main.c then perform commitment. Note that using 'status' command indicates merge is needed:

cvs status main.c

==========

File:main.c Status: Needs Merge

Working revision: 1.1 Sat Oct 9 10:45:22 2010

Repository revision: 1.2 /cvsroot/pr1/main.c,v

cvs update main.c

RCS file: / cvsroot/pr1/main.c,v

retrieving revision 1.1

retrieving revision 1.2

Merging differences between 1.1 and 1.2 into main.c

M main.c

The 'M' means that CVS has automatically merged local version with the latest reversion of repository.

Note: Always run *update* before committing any changes.

8. **Conflict**: if *update* command couldn't merge the files, then a conflict occurs.

   Cvs update main.c

   RCS file: / cvsroot/pr1/main.c,v

   retrieving revision 1.2

   retrieving revision 1.3

   Merging differences between 1.2 and 1.3 into main.c

   rcsmerge: warning: conflicts during merge

   cvs update: conflicts found in limerick.txt

   C main.c

   The 'C' means that CVS tried to automatically merge the two files, but conflict arose. This requires user's action. Conflicts are always indicated like this:

   <<<<<<< local-file

local file text that doesn't jive with repository version

=======

repository version text that doesn't jive with local file

>>>>>>> revision number

Graphical User Interface

1. Windows-based

Any CVS client software could be used to connect to CVS server which is installed on the COBRA server. COBRA server address is 131.202.9.73 and can be connected from anywhere in the university. First step is to use '*check out*' command to get a copy of project into your local machine. Then after adding or editing files, you should '*commit*' changes on server. Don't forget to use other commands such as '*update'.*

a. **TortoiseCVS** (freeware)

Download link: http://www.tortoisecvs.org/download.shtml

By installing this software, some new CVS commands will be added to context menue as displayed in figure 1.

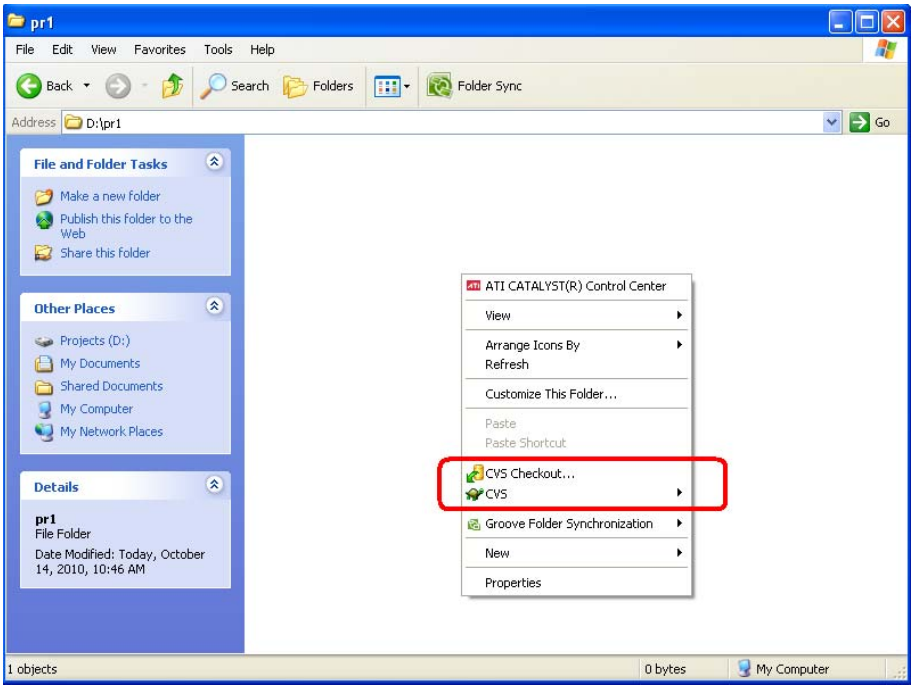


Figure 1 Tortoise CVS is added to context menu

Use *CVS Checkout…* command to download source codes into your local directory. Put your username and other parameters of connection as displayed in figure 2. After checking out project, you can edit or add any other file to the project. When a file is edited, its icon will be changed as displayed in figure 3. The changes can be committed by performing *commit* command.

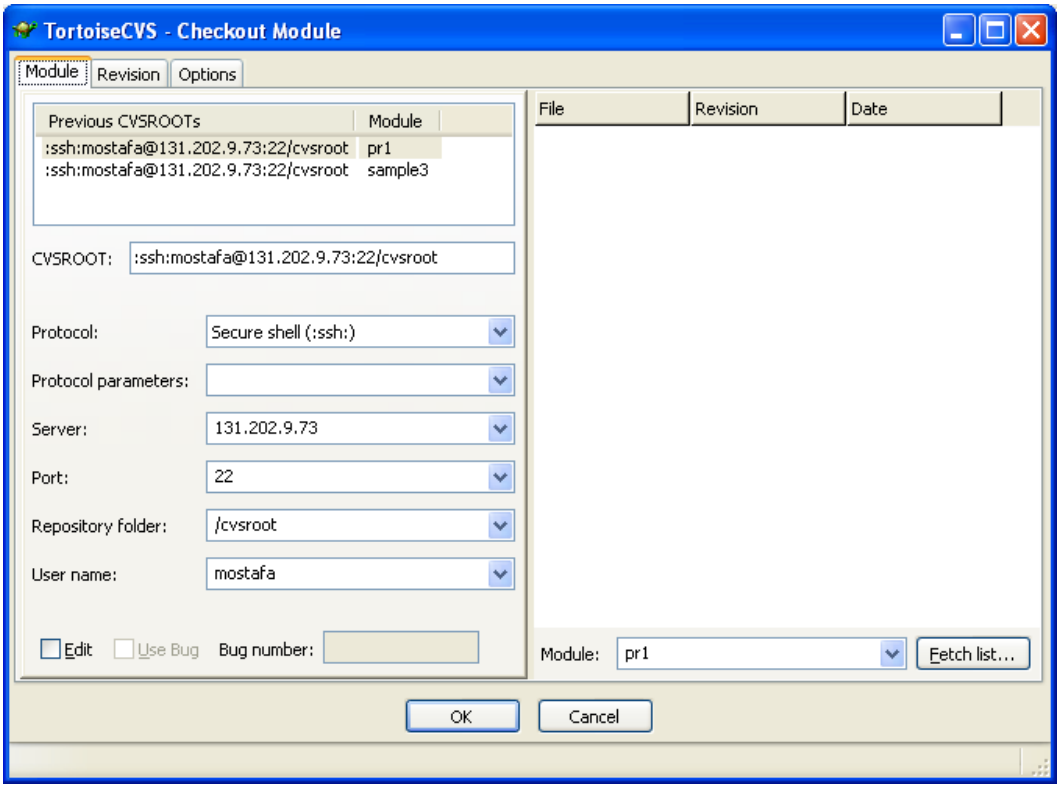


Figure 2 Checkout dialog box

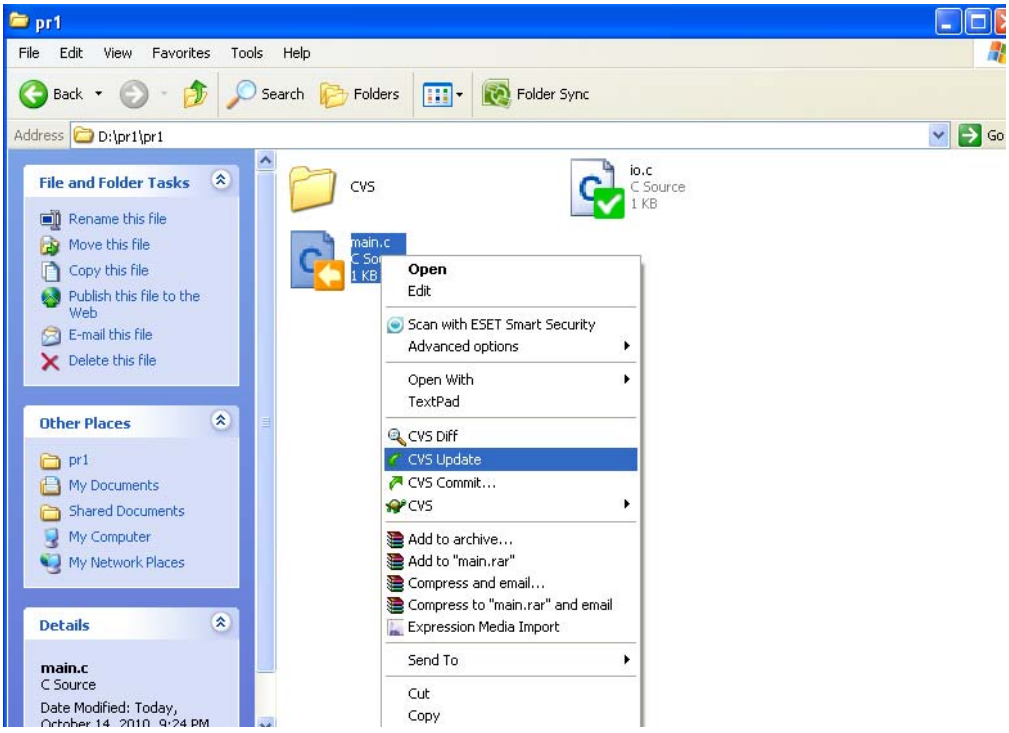


Figure 3 CVS commands

b. **smartCVS** (shareware)

trial version can be downloaded from
http://www.syntevo.com/smartcvs/download.html

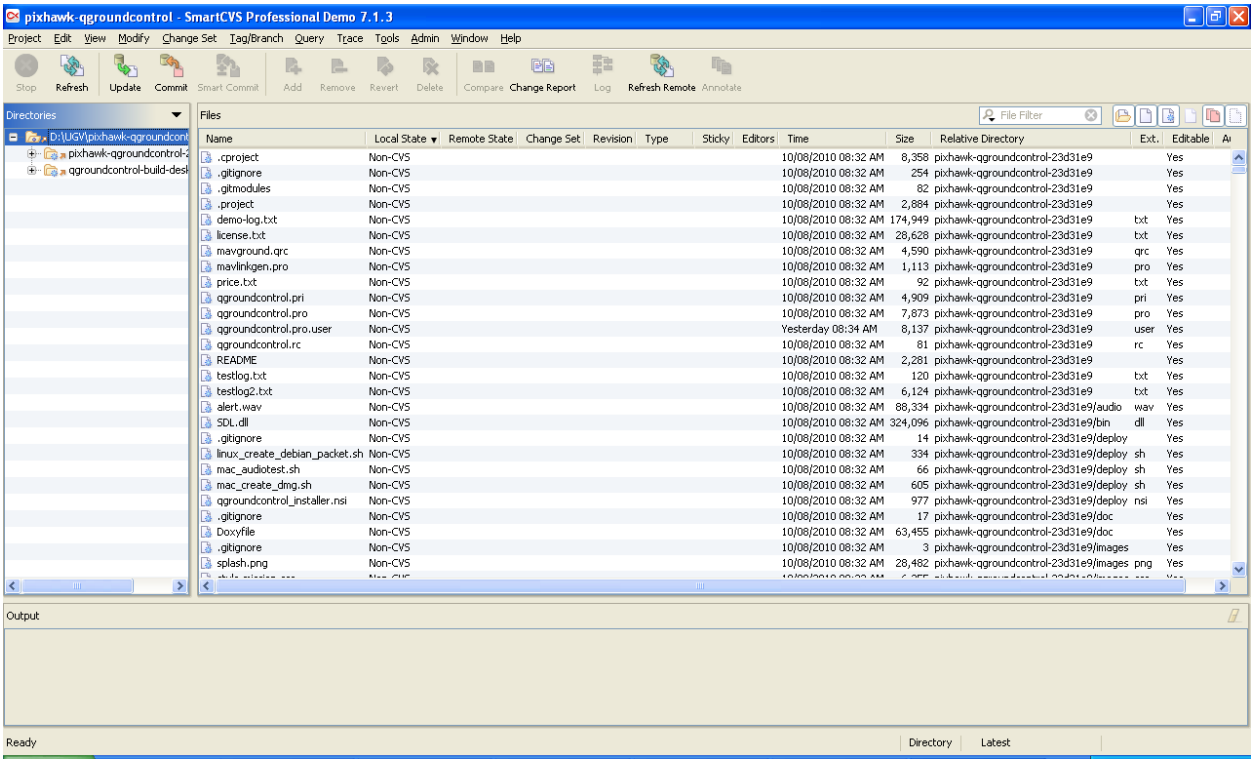


Figure 4 GUI of smartCVS software

To checkout a project, go to *project* menu and click on '*Check Out …'* follow the wizard to connect to the repository and download the project into your local directory. Follow figures 5-13 to download project from the repository on the server.

After these steps, you can change any files and then again by CVS commands like *update, commit, add,* … perform your changes and upload them into the server.

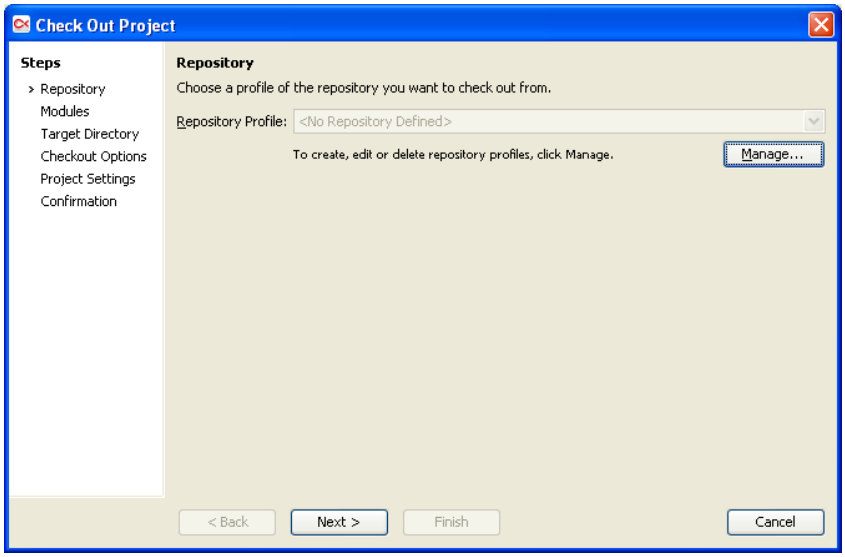


Figure 5 check out wizard, if you have no repository, press *Manage…* button.

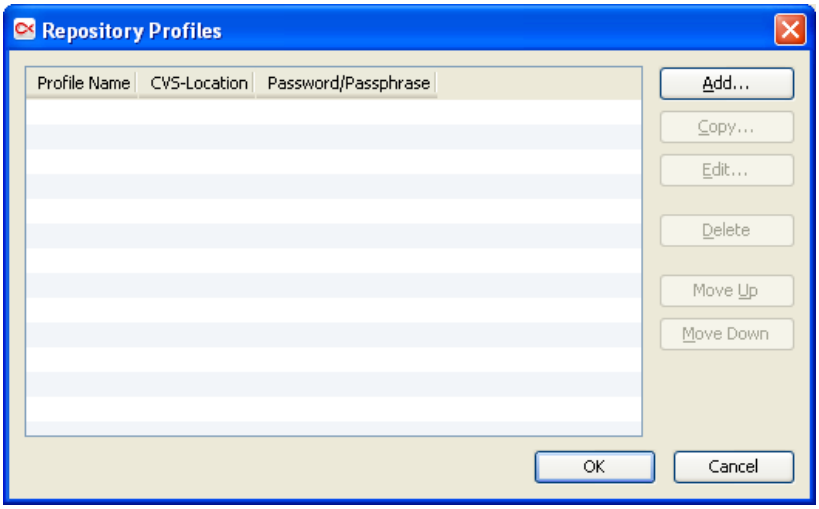


Figure 6 add new repository, if you have no profile, press *Add …* button.

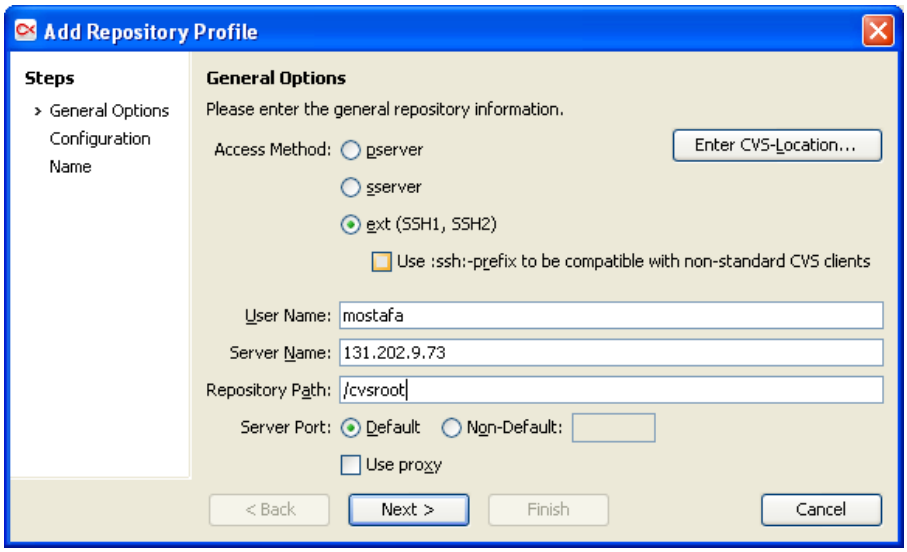


Figure 7 general options of new repository

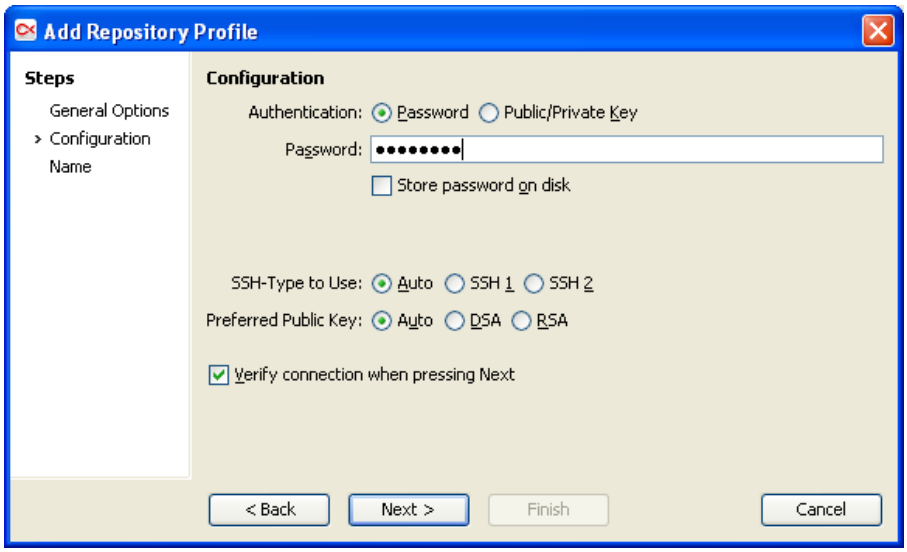


Figure 8 enter your password

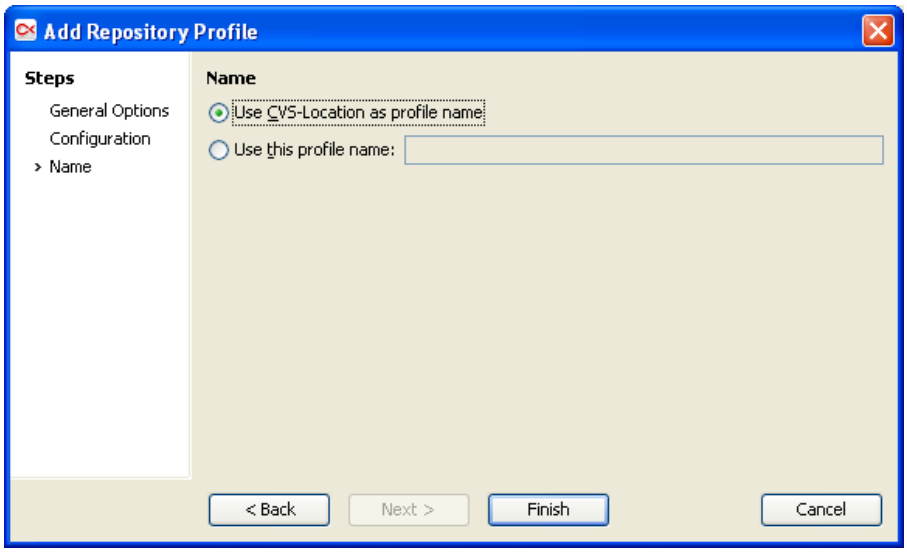
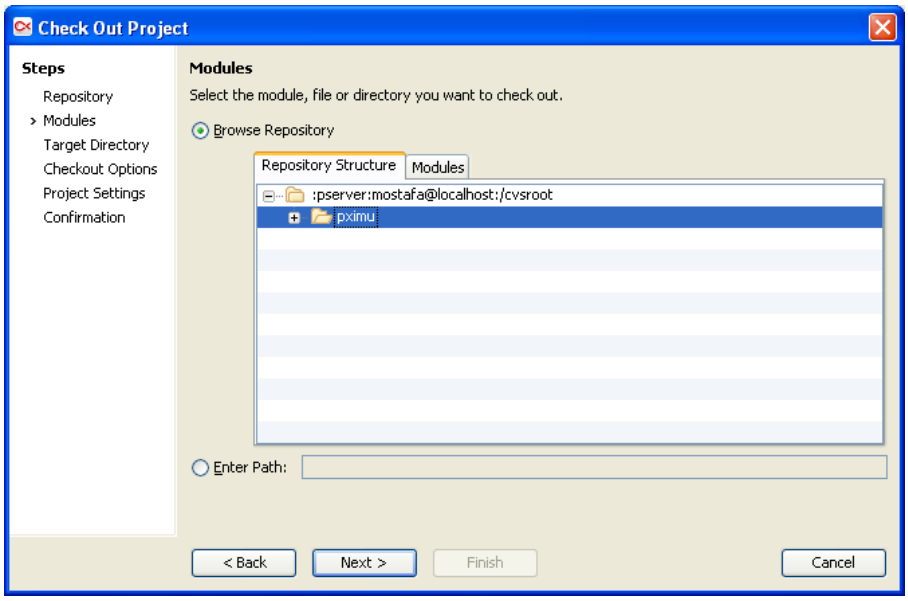


Figure 9 finish this repository profile



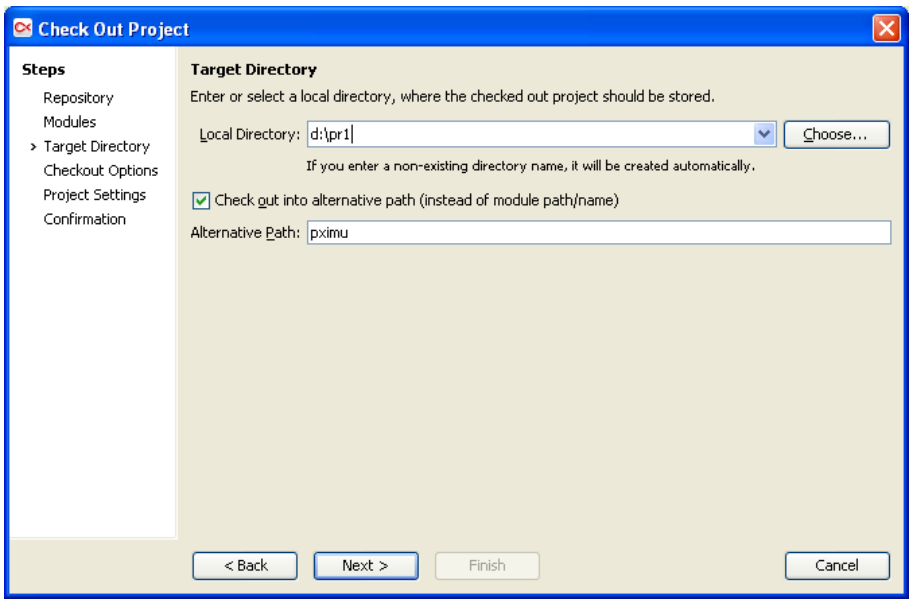


Figure 10 enter your local directory to store project

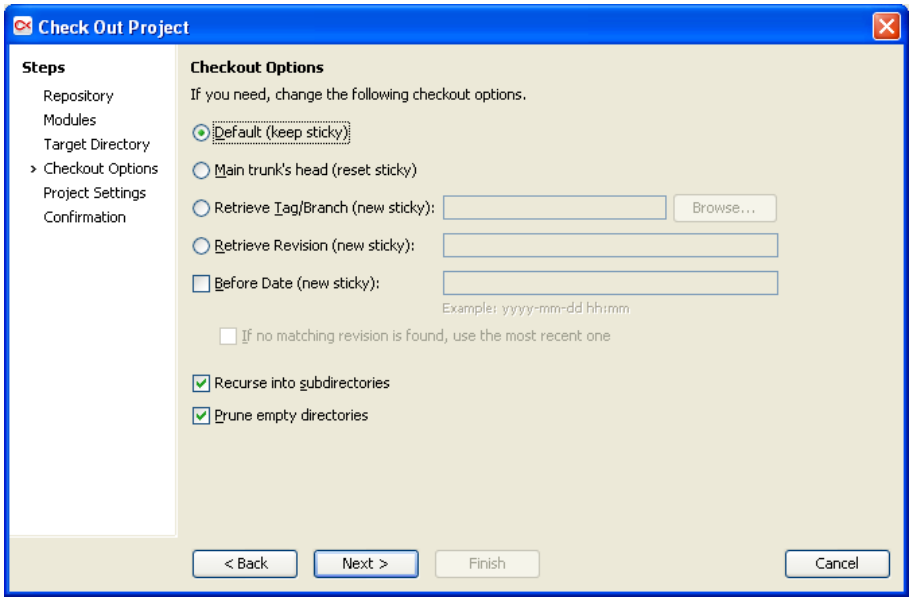


Figure 11 checkout default options

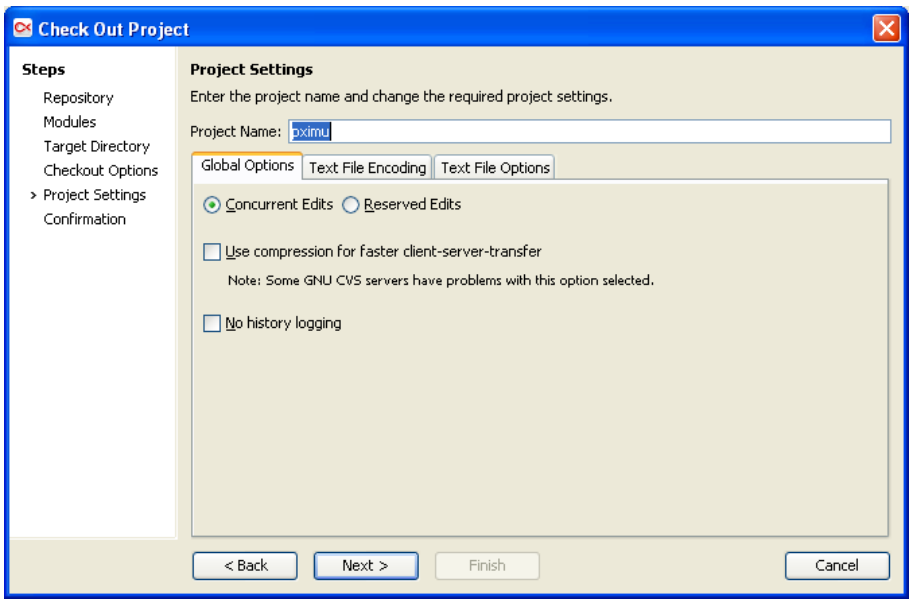


Figure 12 set project name

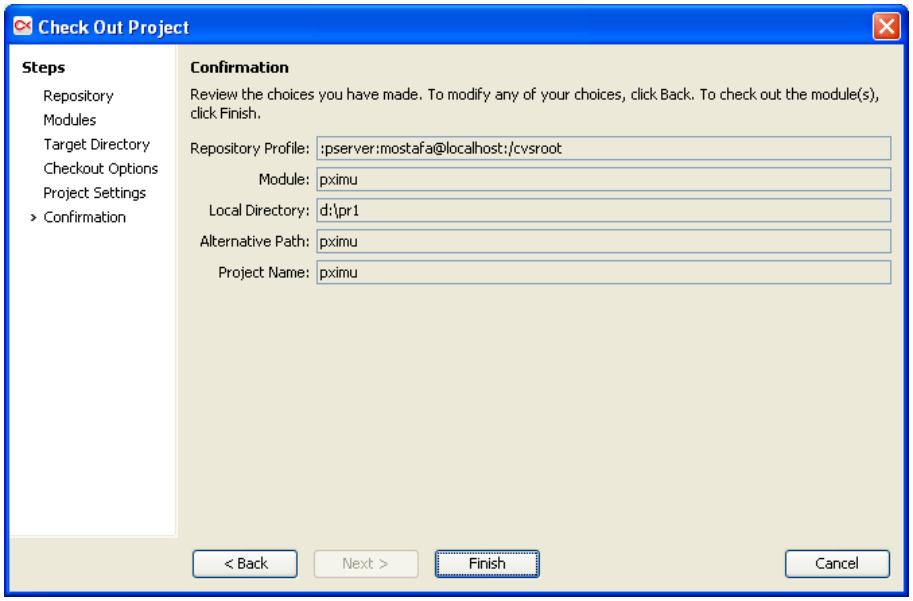


Figure 13 finish the wizard

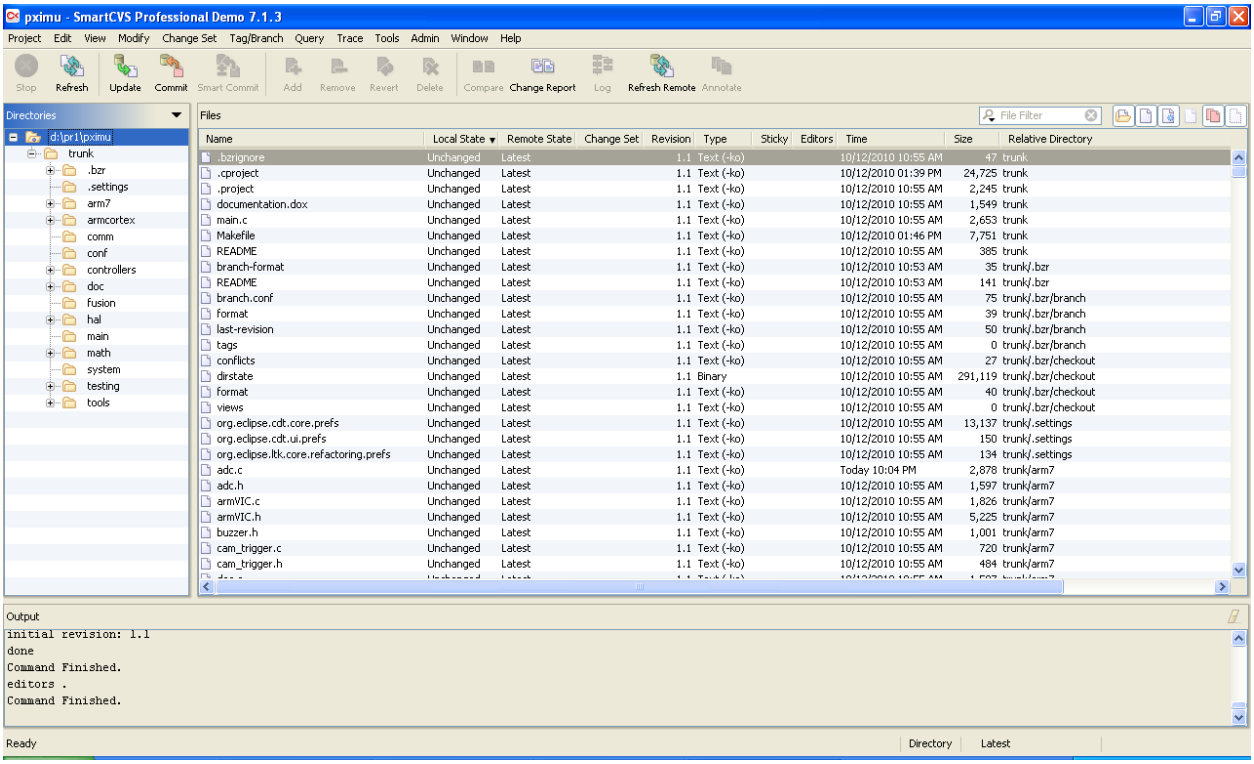


Figure 14 all files will be listed in the main GUI window.

2. Linux-based

http://www.cjlinux.com/software/development/how-to-install-cervisia-in-ubuntu.html

Cervisia is a front-end for the CVS version control system client.

In addition to basic and advanced CVS operations, it provides a convenient graphical interface for viewing, editing, and manipulating files in a CVS repository or working directory. It includes tools designed to ease the use of CVS, such as a log browser, conflict resolver, and changelog editor that checks for incorrect formatting.

This software can be installed on Ubuntu by following command:

sudo apt-get install cervisia

To run this software, type ‘cervisia’ in a command shell.

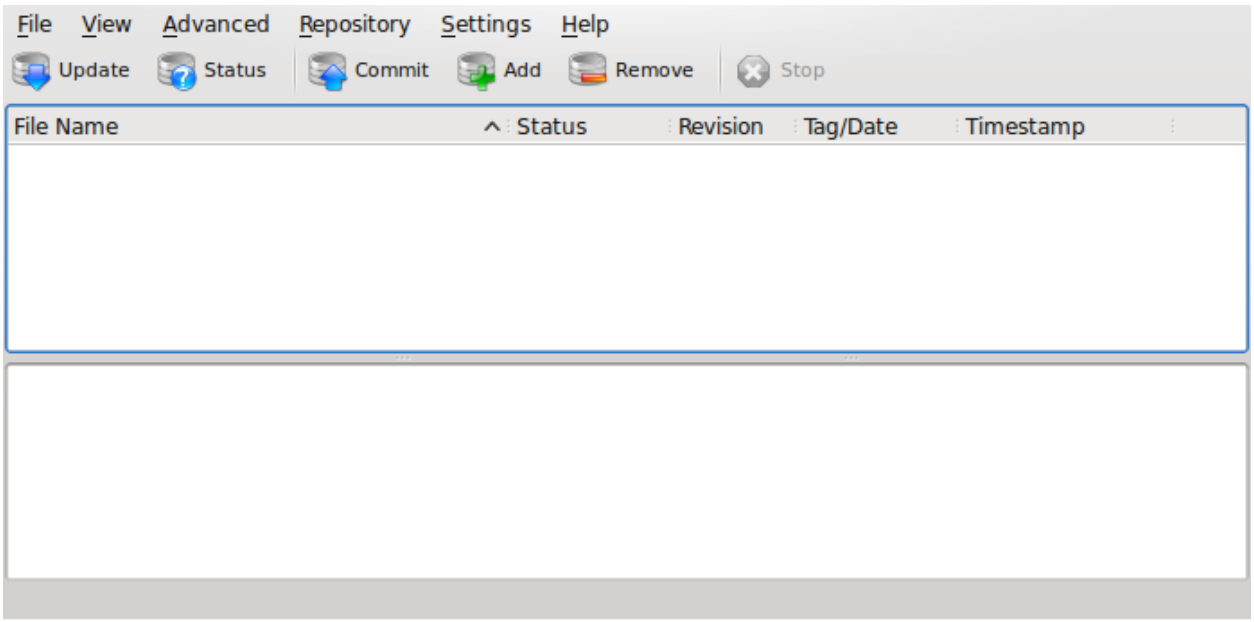


Figure 15 GUI of the cervisia

To import new project, go to *Repository* menu and select *checkout …*

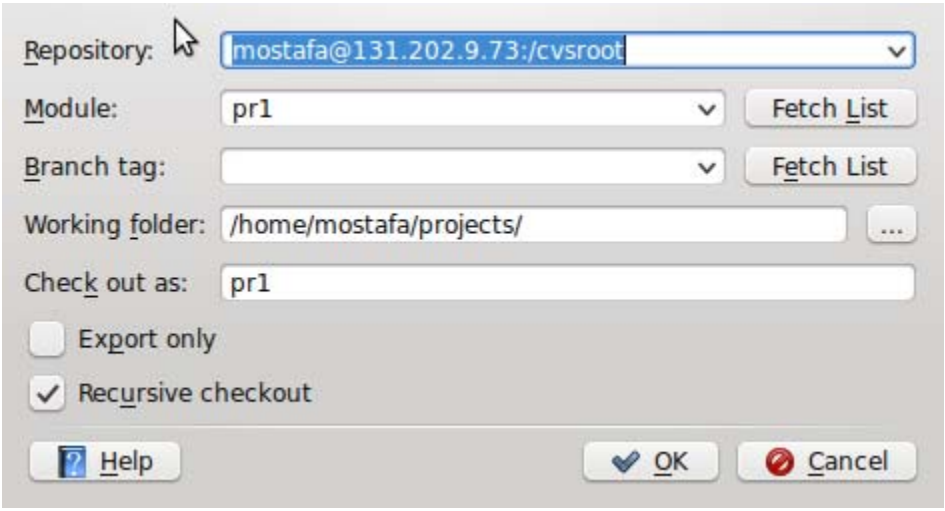


Figure 16 checkout wizard

Files of the project will be listed in the main windows. Changes can be committed easily by pressing '*commit'* button.

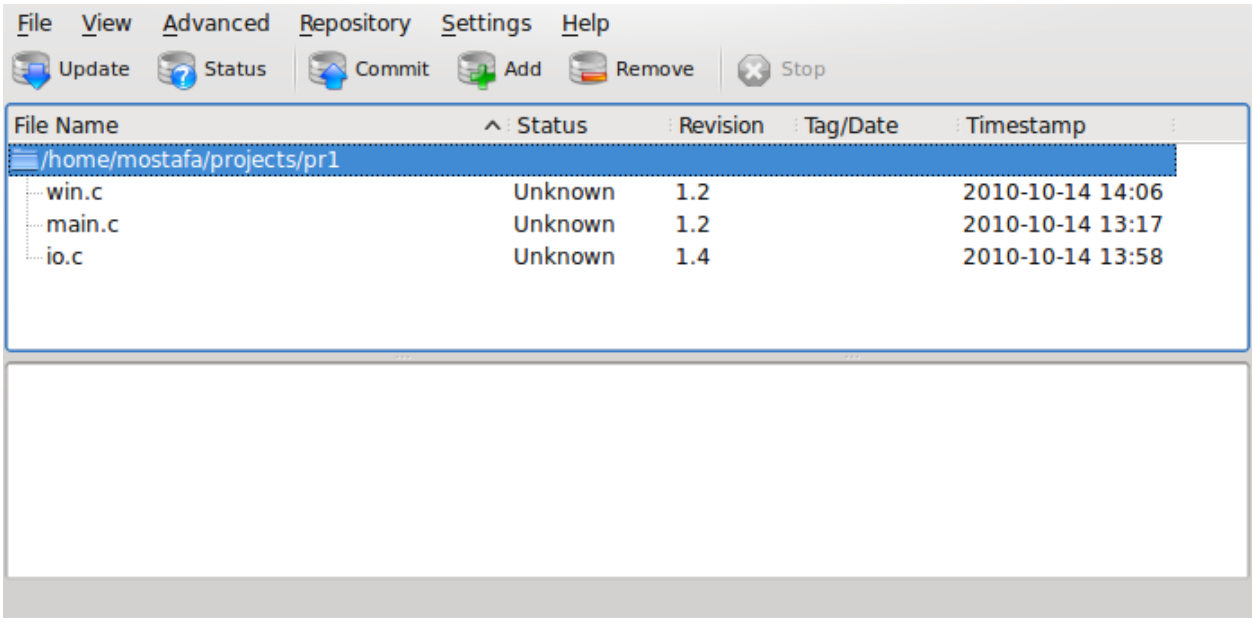


Figure 17 files of the project will be displayed in the main window.