Republican Guide to Linux

Appendix E: Source Control Systems

Typically, when you often operate with text files that require revisions before their release or when you are concerned with traceability of changes in your own files it is a fair indication that you might need to think about source control system for your text files even if these files are not supposed to be used in shared mode with other users.

Windows	Linux
TortoiseSVN is the source control system integrated with Windows Explorer, and is based on SVN source control system earlier introduced with Linux. SVN itself was inspired by CVS source control system used with Unix since 1990 (and later with Linux). The advantage of TortoiseSVN is that Windows repository created with TortoiseSVN can be used as repository when accessing with Linux-based client SVN tools. This Windows/Linux interoperability along with integration with Windows Explorer sounded appealing for me when I opted to use TortoiseSVN for my own projects. One other benefit of TortoiseSVN is that it is able to compare Word files (though not side-by-side, v1.4.5), and works with UNICODE. As with any source control system you should plan a central repository. It is a good idea to place repository on file server or use internet server, and use shared drive if the repository should be accessed from Windows and Linux. In simple cases, you can choose flash drive. To setup SVN repository after running setup program make new folder (SVN for example), right-click and choose Create repository here... . If you'd try to create repository starting from not-empty folder, the TortoiseSVN will be complaining, which means that repository can be created from empty folder only. Once repository is created, you can add files (and	With Linux there are 2 major options[1]. The first one is to use CVS, and the second is to rely on Subversion (or SVN), which inherits major CVS features and claims removing some shortcomings observed in CVS. If circumstances do not require from you to use CVS, the use of SVN might be a better option, especially if you need to track versions of UNICODE files. On the other hand the use of CVS is simpler, more intuitive, and CVS is integrated with KDE by default through Konqueror file manager, even though you can use a separate UI frond-end for CVS called Cervisia (**cervisia**). Cervisia enables you to setup a CVS repository and run basic tasks such as commit, checkout, update, adding files, viewing version differences without running these tasks in command-line interpreter. Sure you can use console (**konsole**), but unless you use CVS everyday keeping in mind its commands might not be a reasonable and easy exercise (see the list of relative commands with: # **cvs** --help-options or enter **man**:**cvs** in Konqueror's address bar). The next steps show you how to setup an empty CVS repository from scratch so that files and folders can be added later. It is a good idea to setup repository on file or internet server, however, for simplicity, you can use flash-drive for tests: • Run Cervisia (**cervisia**), and create repository cvs_repo from menu option Repository/Create...

folders) to the SVN repository. You do this with Repo-browser, which can be invoked by right-click on repository, choosing TortoiseSVN, and then Repo-browser.

Now, when you need to work with repository files locally, you first make a checkout (right-click on repository and choose SVN Checkout... or checkout from repo-browser). For the destination of the local copy choose any location such as MyDocuments/LocalSVNCopy. TortoiseSVN enables right-click menu for the folder where you made a checkout. The new context-menu items are also available for local files. Notice that TortoiseSVN adds red exclamation mark to the left bottom of an icon of locally edited file, and green mark if a local file was not edited since last checkout. The same meaning have green and red marks for folder icons. In fact, it is convenient to identify folder as local SVN copy by these folder marks.

Among menu items that SVN adds for locally edited file, the most useful is Diff option, which makes possible to see differences between local and server (repository) versions. Don't forget to update repo files if you make changes to the local copy (right-click on file and choose SVN Commit...). The accidental destruction of the local copy can cause no harm if you did not forget to make commits after updates of the local files. To restore local copy you simple make a checkout from the relative repo.

When new file is added to the repositoy, you can load it to the local directory by right-click on local project folder and choosing SVN Update.

If you have programming experience with Microsoft development tools, you can jumpstart using TortoiseSVN having in mind that mode of operations with TortoiseSVN is based on paradigm used with other source control systems. Take a look on Visual Source Safe window, which is the Windows source control system provided by Microsoft basically for version control of source files while programming. The TortoiseSVN's analog of Visual Source Safe Explorer is already mentioned Repository Browser.

(enter an appropriate path such as /media/disk/cvs_repo to your flash-drive). In the result, the default structure is created under cvs_repo folder (at this point there is only default CSVROOT folder with CVS administrative files, which should not be ever edited manually).

- Create a working directory such as cvs_client and import a project (module) you are going to use, for example TODO (you can specify any module name such as TODO even though it still does not exist under cvs_client) from cvs_client with Cervisia's Repository/Import... This step is an equivalent of creating an new (and empty) project in CVS repository. The fields "Vendor Tag" and "Release Tag" do not matter in fact albeit mandatory (CVS keeps these fields for historical reasons, you can use company name for "Vendor Tag" and "start", without quotes, for "Release Tag"). Notice that under CVS a project is called module, and modules are located directly under repository folder (cvs_repo) that is on the same level with default CSVROOT folder.
- Create a working copy of TODO project (module) by making checkout with Cervisia's Repository/Checkout... This step includes initializing of working copy with CVS administrative files (folder CVS under project name TODO of cvs_client working directory).
- At this point the repository is initialized, has empty project (module) TODO, and also you have its working copy (folder TODO under cvs_client). Now you can add files to the repository. Firstly, you simply copy necessary files to cvs_client/TODO. Then right-click on TODO folder and choose Open With/Cervisia. This opens Cervisia within Konqueror file manager. Now right-click on local file such as TODO.txt and choose Add to Repository... You need also make Commit... [2] in the same menu to accomplish adding TODO.txt to the repository.

With Linux, SVN has become de-facto standard for users and developers who want to obtain latest versions of source code for a package to build it on local machine albeit obtaining source with FTP is no less practical. Frequently, a user needs a few SVN commands such as checkout, which can be effectively run (and bookmarked within console) from console (**konsole**), without invoking UI-enabled front ends. For

this reason, I provide here in a nutshell basic steps necessary to setup SVN repository without UI-enabled tools (the order of steps reflects the steps with CVS repository setup). As before you can use for tests flash-drive:

- Create a repository such as svn_repo:

  # **svnadmin create** /media/disk/svn_repo

  At this point SVN creates a structure with administrative files.
- Create working directory, where you want to work with local copy:

  # **mkdir** /media/disk/svn_client

  Or, use Konqueror to create an empty folder svn_client.
- Create a new project under SVN repository. This means, like with CVS, to import an empty local directory like svn_client. The custom name TODO of new project is set in command line:

  # **svn import** /media/disk/svn_client file:///media/disk/svn_repo/TODO **-m** 'initialization'

  Notice that **-m** (log message), providing a description of revision, is mandatory.
- As with CVS, at this point SVN repo is initialized and has an empty TODO project, you need to make a checkout to start working with project locally:

  # **svn checkout** file:///media/disk/svn_repo /media/disk/svn_client

  This creates a local copy with .svn folder under svn_client containing administrative files. In general, the local folder (svn_client in the sample) is created if it was not set in command line.
- As with CVS you actually start working with repository by adding and committing files. First, add to the working directory svn_client a file, for example TODO.txt, and add it to the repository. Since making **add** does not actually adds a file to the repository you need, in fact, execute two commands (**add** and **commit**):

svn add /media/disk/svn_client/todo/TODO.txt
svn commit /media/disk/svn_client **-m** 'commit test'

As before, **-m** (log message), providing a description of an action, is mandatory.

With KDE and GNOME come KDESvn (**kdesvn**) and RapidSVN (**rapidsvn**) programs relatively that enable you to accomplish source control tasks within graphical interface avoiding the use of console commands (you still need to create repository in console when using RapidSVN). These programs represent UI front-ends to Subversion (SVN commands).

The sequence of steps to create a new repository and add to it a file are the same as described when using console:

- Create new repository: run **kdesvn**, go to File/Subversion Admin and choose Create and open new repository (uncheck Create main folders since for simple tasks you probably don't need standard Subversion's structure: trunk, branches, tags).
- Create new project by adding new folder TODO from Subversion/General/New Folder. Notice that KDESvn hides the use of **import** command, which was used with console SVN commands to create TODO project.
- Create a local copy of new project by right-click on TODO and making checkout to local directory.
- Add a file to repository: make a file TODO.txt under TODO folder of local copy, right-click on TODO folder and choose Open With/kdesvn, then right-click on TODO.txt file and choose "Add selected files/dirs". As before, you still need to Commit in the same menu to actually add file to the repository.

The pictures used above were obtained with Cervisia v2.4.9 and KDESvn v0.11.2 [3].

1. With Fedora 7 both CVS (**cvs**-1.11.22-9.fc7.src.rpm) and Subversion (**subversion**-1.4.3-4.src.rpm) packages are installed by default. You can verify this either by looking into *install.log* in ROOT directory or running # **rpm** -q cvs and # **rpm** -q subversion (or # **rpm** -qi ... for more details). With Red Hat 9 CVS is also installed by default (**cvs**-1.11.2-10.i386.rpm), and one can install Subversion (**subversion**-0.17.1-4503.0.i386.rpm) from the CD#3.

2. By default ROOT user is not allowed to commit changes in CVS. If you try, you'll have "root is not allowed to commit files" error message. You can another account or, if you prefer to work with CVS under ROOT, you need to recompile CVS with --enable-rootcommit switch. Here are details:
    - Download latest stable CVS source from the site (it was **cvs**-1.11.23.tar.gz at the time of writing).
    - Copy into /usr/local, unzip and untar the source:

      # **gzip** -d '/usr/local/cvs-1.11.23.tar.gz'
      # **tar** xvf '/usr/local/cvs-1.11.23.tar'
    - Compile with --enable-rootcommit option and make (/usr/local/cvs-1.11.23 must be current directory):

      # ./configure --enable-rootcommit
      # **make**
      # **make install**

    Notice that you can check the current version with # **cvs** --version (1.11.23 if you did described steps).
3. Cervisia, KDESvn, Rapid SVN do not come with Fedora 7 by default (CVS v1.11.22 is installed by default). In fact, Cervisia (**cervisia**) is a part of **kdesdk** package. To download these packages make sure that you are connected to the internet and use **yum** to download and install:

    # **yum install** kdesdk
    # **yum install** kdesvn
    # **yum install** rapidsvn