Last updated March 6, 2002 by Ulrich Moeller osFree updated October 11, 2003 by Yuri Prokushev

Note: This document in process of update for osFree SVN instead of Netlabs.org CVS.

ABOUT THIS DOCUMENT

This is for people who would like to access the osFree source code. If you have never written a program before, you will probably not be interested in this document.

INTRODUCTION TO SVN

SVN stands for “S…. V…. N….”. It is a tool which allows several developers to work on the same source code at the same time.

For those of you that have never dealt with anything like this… just imagine a word processor document on a shared network directory that several people work on at the same time. There has to be some coordination of all the changes going on and checks whether the changes conflict with each other.

SVN does exactly that, except that it can handle large directory trees with many files of any type – text or binary.

The concept of a client/server SVN setup is that the server maintains a “repository” with the entire source code, including all changes that were ever made. Each developer then creates a local copy of the sources on his hard disk. So each developer can safely work on the local sources until he thinks that everyone else should get the changes (once they're halfway stable), and can then put the changes on the server, and all other developers can then update their local copy.

While your local copy will always be at a certain code/revision level (and only updated from the server if you request this), the server maintains all changes that were ever made. You can, for example, even retrieve the sources in the state were in in the middle of 1999, if you want.

Maybe the nicest thing about SVN is that with text files, it can merge code changes. SVN has GNU “diff” built in so even if two people work on the same source file, SVN in most cases can automatically keep the changes in sync. See “Terminology” below.

Netlabs has set up a CVS server on which all the Netlabs projects reside. This document should give you all the information about how to access that repository.

SETUP

As with most Unix programs, SVN expects you to have a “home” directory. If you don't have one, create an empty directory somewhere and set the HOME environment variable to that directory.

In any case, I strongly recommend to create a file in $(HOME) called ”.svnrc” and add put a single line

cvs -z9

in there to enable maximum compression during transfers. This greatly speeds up things if you have a slow internet connection. Otherwise CVS will do all network transfers with uncompressed data, which can be terribly slow.

Secondly, the most practical way to use CVS is to have a common parent directory for all code repositories (i.e. projects). So create one (for example, “F:\cvs”), which will become the parent directory for all CVS projects (e.g. “F:\cvs\osFree”, “F:\cvs\warpin”, “F:\cvs\xworkplace”). If you later check out repositories from sources other than Netlabs (e.g. from SourceForge), they can go into this root as well.

Finally, CVS needs you to have a user name and a password. If you don't have one yet, use “guest” and “readonly”, which will give you read-only access to all Netlabs projects. (Other servers will have different conventions here.)

Set the current project environment. This is done via another environment variable, so for XWorkplace, set:

```
CVSROOT=:pserver:guest@http://www.netlabs.org/netlabs.cvs/xworkplace
```

For WarpIN, use:

```
CVSROOT=:pserver:guest@http://www.netlabs.org/netlabs.cvs/warpin
```

For the “osFree” use:

```
CVSROOT=:pserver:guest@http://www.netlabs.org/netlabs.cvs/osfree
```

and so on. Currently, Netlabs holds over a dozen projects which can all be retrieved that way.

NOTE: The CVSROOT path was changed in February 2002 for all Netlabs projects. From your old CVSROOT variables, simply remove the “e:” before ”/netlabs.cvs”. You will need to run “cvs login” again to have CVS pick up the change (see below).

Create a local subdirectory in your CVS root directory with the same name (e.g. (“F:\cvs\osFree”). Change to that directory.

Set USER=guest in your environment. Fire up your internet connection. To log onto a project, use the “cvs login” command. Enter “readonly” when you are prompted for the password. (Note that you will need to enter “cvs login” separately for each repository. CVS will then store the data in $(HOME)\.cvspass so you only need to do this once per project.)

You are now ready to download (“check out”) a project. See section 5 below.

SVN TERMINOLOGY

The following terms should be known when working with SVN (or even reading the documentation):

- “Repository” is the unit CVS uses for managing projects. For example, “xworkplace”, “warpin”, and “xwphelpers” are three repositories on the Netlabs CVS server. Repositories are mirrored

locally in a subdirectory of your CVS root directory, if you followed the above setup.
- Repositories are usually equivalent to projects. However, repositories are just a directory on the server and can be created in any way. For example, we created the “xwphelpers” repository for code that is shared between XWorkplace and WarpIN.
- “Checkout”: “Checking out” code means that you retrieve (or update) a local copy of files in a repository. You can check out an entire repository or single files from it.
- To check out code, use the “cvs checkout” command. Details follow in the next section.
- “Commit”: If you have write access to the CVS server (which requires that you have been given a user name and password by the Netlabs admin, Adrian Gschwend), you can change the repository by “committing” your local copy back onto the server. Once the code has been committed, all other developers can check it out again and will have the updates.
- To commit code to the server, use the “cvs commit” command. See the CVS reference for details. Again, this requires that you have been granted write access by the server administrator.
- Usually, with most projects, only a limited number of people have write access to the server, for obvious reasons. But this is a “project policy” issue and not a CVS restriction.
- “Merge”: This can happen if you have checked out your local copy of a repository already and do another “checkout” later.

If SVN then finds that a file has been changed on the server and your local copy needs to be updated, it “merges” all changes into the local copy. That is, it does not simply overwrite the local copy with the new version on the server, but will compare (diff) the two files and patch the local copy! As a result, you will not lose changes you made to your local copy when you check out updates.

CVS has GNU “diff” built in for this, so this is pretty powerful.

This happens automatically if you do “cvs checkout” after your initial checkout. Note that you can also use “cvs update”; the difference is that “update” will not retrieve new files, but only update existing local files. See the CVS reference for details.

- “Conflicts”: This happens if CVS cannot successfully merge changes into your local copy. Most frequently, this occurs when two developers have modified the same lines in the same source file. Of course, CVS cannot resolve these conflicts by itself, so it will put two versions into the file marked with special characters. (With C files, this will cause compilation to fail, which is intentional so that you have to manually resolve the conflict.)

```
    See the CVS reference for details.
```

CHECKING OUT THE CODE

Change to the osFree root directory (e.g. “F:\cvs\osFree”) set the osFree environment as described above (HOME, USER, CVSROOT), and log on if you haven't logged on yet.

The command for getting source from the server is “cvs checkout”. With checkout, you need to specify the module to check out. For the purpose of this explanation, we will consider a “module” a directory name. Note that CVS normally operates on directories recursively! So if you use

cvs checkout .

(mind the dot) from the XWorkplace root directory, this will retrieve the entire code tree and create new local files if they don't exist yet. This can take a while for bigger projects. Again, use maximum

compression (see section 3).

“cvs checkout” allows you to specify file masks as well. For example, if you type

cvs checkout src/shared/*.c

only those files are checked out. Mind the forward slashes.

By contrast, “cvs update” only updates local files which have been changed on the server. It does not create new files. Since I frequently create new files on the server, using “checkout” will probably be the better general choice.

You will see that CVS puts out lots of information about the files that were worked on. A typical CVS output from “update” or “checkout” looks like this:

```
? doc/progref.ipf
M makefile
M include/build.h
M include/filesys/program.h
M src/classes/xcenter.c
```

I strongly recommend redirecting standard output when using “cvs update” or “checkout” so you know what CVS has done, since these lines scroll of the screen very quickly. You can simply use “cvs checkout . > checkout.log”. An even better choice is “tee”, another GNU program, which prints the output to the screen and logs it in a file at the same time.

In each such line, the first letter tells you what CVS has done to the file:

?: CVS does not know the file. It is in your local tree, but not in the repository. This happens if you have added files to your local copy yourself.

U: This is the most frequent one. It means that CVS has “U”pdated a local file with a new copy from the server and occurs during “checkout” or “update”.

P: Similar to “U”, except that the server has only sent a “P”atch instead of the entire file. The result isn't any different from “U”; the file has been updated.

M: This means that your local file was “M”odified before the update. You get this no matter if the file was updated (i.e. changes were merged successfully) or not.

C: This happens if merge failed and a “C”onflict occured. You need to manually edit all files which have been marked with “C”; if they are C source files, they will no longer compile.

R: A local file has been removed because it is no longer in the repository (i.e. has been removed by someone with write access).

Discussion