NetBSD / solarpack synchronizations

Author: Marc Herbert
Revision: 1.6

Contents

TODO: standardize syntax of solarpack tags

1 References documentation

Documentations on `cvs import` or: tracking third-party sources

- in CVS manual (info)
  - offline: info cvs, go to node: import
  - online: on cvshome.org
- in CVS book
  - offline: try file:///usr/share/doc/cvsbook/cvsbook_5.html#SEC93 or similar
  - online: on cvs.red-bean.com
- in CVS FAQ
  - online: in the old but good CVS FAQ

2 NetBSD releases

- NetBSD Release map
- NetBSD Release history

NetBSD does seldom uses tags on their `pkgsrc` source tree, as you can check on this file for instance.

This raises an open issue: do we synchronize more often than their tags, and if yes when?

3 HOWTO

Never use the CVSROOT variable, it's confusing and of little use. Instead define the SOLARCVS and NETBSDCVS variables, and use the explicit -d option on the very seldom cases where cvs is not able to find out the repository address by itself.

Check the mirror list and use a mirror close to your location.

3.1 First import (or: how solarpack was born)

– initialization of our repository:

  ```
  boss$ cvs -d $SOLARCVS init
  ```

– get ("export") the official upstream version that will be merged. Use the -kv option, so the CVS server will strip all $keyword: value$, leaving only the value in sources, and so version numbers from NetBSD won't be mixed with solarpack version numbers. See the short CVS keywords demystification below:

  ```
  boss$ cd /tmp
  boss$ cvs -d $NETBSDCVS export -kv -r netbsd-1-6-PATCH001-RC1 pkgsrc
  ```

– import it into our repository, into the special vendor branch (called "upstream"):

  ```
  boss$ cd /tmp/pkgsrc
  boss$ cvs -d $SOLARCVS import solarpkgsrc upstream netbsd-1-6-PATCH001-RC1
  ```

  create a working directory, and start to work:

  ```
  developer$ cd Solarwork
  developer$ cvs -d $SOLARCVS checkout solarpkgsrc
  ```

  [edit, compile, test, commit, repeat] See the solarpack developer documentation

3.2 Synchronization with NetBSD

Time to synchronize/merge code with a new NetBSD upstream version

Say synchronisation with netbsd-1-6-PATCH003 for instance.

– Warn all solarpack developers that a merge will occur soon : they must stabilize & commit everything now (better to solve merging issues one at a time).

  [wait a few days until they are done]

– Freeze: order other developers to stop developing (or at least to stop committing for a while). Only the boss is working from now on, until further notice. The boss can easily check that no one is still working:

  ```
  boss$ cvs update -n
  ```

– tag the current version (there is never too many tags):

  ```
  boss$ cd Solarwork/solarpkgsrc
  boss$ cvs tag before_1-6-PATCH003_merge .
  ```

– get the new official upstream version that will be merged:

```
boss$ cd /tmp
boss$ cvs -d $NETBSDCVS export -kv -r netbsd-1-6-PATCH003 pkgsrc
```

– import it in our repository, on the special vendor branch (critical: use same vendor tag across imports):

```
boss$ cd /tmp/pkgsrc
boss$ cvs -d $SOLARCVS import solarpkgsrc upstream netbsd-1-6-PATCH003 \
            | tee ~ /import3.log
```

– split the merging work among a small team of core developers. Note: "cvs checkout" is also able to apply patches in working dir, similar to "cvs update". See CVS references above:

```
coredeveloper$ cd Solarwork
coredeveloper$ cvs -d $SOLARCVS checkout -jnetbsd-1-6-PATCH001-RC1 \
    -jnetbsd-1-6-PATCH003 solarpkgsrc
```

– (try to) compile and make it work

[edit, compile, test, repeat]

– at this point, make the minimum changes so it hardly works: it's NOT the moment to start to rewrite everything because this merge gave you a new great idea.

incrementally commit the results of the merge on the main development branch:

```
coredeveloper$ cd Solarwork/solarpkgsrc/my_part_of_the_job
coredeveloper$ cvs diff -u
coredeveloper$ cvs commit -m 'finished synchronization with upstream 1.6 \
                    patch 3; had to modify blabla...'
```

– when the core team has finished committing into the main branch all what the new upstream version brought, freeze and proudly tag the result:

```
boss$ cvs tag after_[successful_]1-6-PATCH003_merge .
```

– tell all other developers they may `cvs update` and restart their normal life

4 CVS keywords demystification

Some basic facts, gathered from CVS documentations.

CVS keywords expansion (or *substitution*) relies on the CVS **server**. The CVS server code is historically derived from and compatible with RCS; facts below are relevant to both CVS and RCS.

When going **to** the repository (`import`, `checkin`,...) the files are *never* modified, no keyword substitution of any kind is ever performed. However, you may specify a default expansion mode for future operations.

Keywords substitutions are performed **by the server**, when going **from** the repository (`checkout`, `export`, `diff`,...),

There are four different modes for keywords expansion: `-ko`, `-kkv`, `-kk` and `-kv`.

The expansion mode can be specified in three different ways, in decreasing priority:

1. on the command line, by the client, with a `-kXX` option
2. by a per file default setting located in the client working directory. This default is set by some command line invocations, for instance by `checkout` but not by `diff`. The `-kXX` option is *sticky* in this case.
3. by a per file default setting located on the server, set at creation time (i.e., by `cvs import` or `cvs add`)

`-kkv` is the default mode when none of the above is ever specified.

The four expansion modes are easier to memorize than it may appear at first sight.

`-ko`: perform no expansion and get the **original** file, strictly identical to what it was at `import/commit` time

`-kkv`: default: get both keyword and value. Example: `$ Revision:  3.1415 $`

`-kk`: get only keyword. Example: `$ Revision $`

`-kv`: get only value, thus freezing forever the file. Example: `3.1415`

Some CVS servers (notably official NetBSD CVS servers) are patched *and* configured to recognize custom keywords (`$NetBSD$`) and treat them just like another keyword.