CVSup Advanced Points

Salvo Bartolotta

bartequi@neomedia.it

$FreeBSD: doc/en_US.ISO8859-1/articles/cvsup-advanced/article.sgml,v 1.15 2006/08/29 07:38:22 blackend Exp $

The present article assumes a basic understanding of **CVSup** operation. It documents several delicate issues connected with source synchronization via **CVSup**, viz. effective solutions to the problem of stale files as well as special source updating cases; which issues are likely to cause apparently inexplicable troubles.

Table of Contents

1 Preface

This document is the fruit of the author's attempts to fully understand the niceties of **CVSup** & source updating. :-) While the author has made every effort to make these pages as informative and correct as possible, he is only human and may have made all sorts of typos, mistakes, etc. He will be very grateful for any comments and/or suggestions you send to his e-mail address, <`bartequi@neomedia.it`>.

2 Introduction

If you have visited John Polstra's site (http://www.polstra.com/) and read his FAQ (http://www.polstra.com/projects/freeware/CVSup/faq.html), you may have noticed Question 12 & 13.

When updating any collection of sources (eg `/usr/ports`), cvsup(1) makes use of the related checkouts file in order to perform the updating process in the most efficient and correct way. In this example (`/usr/ports`), the related checkouts file is `/usr/sup/ports-all/checkouts.cvs:.` if your base is `/usr`.

A checkouts file contains information on the current status of your sources—in a way, a sort of "photograph". This significant information enables `cvsup` to retrieve updates most effectively. Further, and maybe more important, it enables `cvsup` to correctly manage your sources by locally deleting any files no longer present in the repository, thus leaving no stale files on your system. In fact, without a checkouts file, `cvsup` would *not* know which files your collection was composed of (cf cvsup(1) and the fallback method for details); as a result, it could not delete on your system those files no longer present in the repository. They would remain on your system (stale files), and might cause you subtle build failures or other trouble. For example, this problem is likely to occur if you first update your ports collection several weeks after you got your installation CD-ROMs.

It is therefore recommended that you adopt the two-step procedure outlined in the **CVSup** FAQ (cf Q12, Q13); in subsequent sections, you will be given interesting and instructive concrete examples.

3 A useful python script: `cvsupchk`

Alternatively, in order to examine your sources for inconsistencies, you may wish to utilize the `cvsupchk` python script; which script is currently found in `/usr/ports/net/cvsup/work/cvsup-16.1/contrib/cvsupchk`, together with a nice `README`. Prerequisites:

1. `/usr/ports/net/cvsup #` **`make extract`**
2. python (also found in the ports collection :-))
3. a checkouts file for your collection of sources.

If you are updating your sources for the very first time, of course you do not have a checkouts file. After installing python and updating your sources (eg `/usr/ports`), you can check them thus:

```
% /path/to/cvsupchk -d /usr -c /usr/sup/ports-all/checkouts.cvs:. | more
```

If you want to check your RELENG_4 sources:

```
% /path/to/cvsupchk -d /usr -c /usr/sup/src-all/checkouts.cvs:RELENG_4 | more
```

In each case, `cvsupchk` will inspect your sources for inconsistencies by utilizing the information contained in the related checkouts file. Such anomalies as deleted files being present (aka stale files), missing checked-out files, extra RCS files, and dead directories will be printed to standard output.

In the next section, we will provide important, typical examples of source updating; which examples will show you the role of checkouts files and the dangers of negligent source management.

4 Examples of more advanced source management

4.1 How to safely change tags when updating `src-all`

If you specify eg `tag=A` in your `supfile`, `cvsup` will create a checkouts file called `checkouts.cvs:A`: for instance, if `tag=RELENG_4`, a checkouts file called `checkouts.cvs:RELENG_4` is generated. This file will be used

to retrieve and/or store information identifying your 4-STABLE sources.

When tracking `src-all`, if you wish to pass from `tag=A` to `tag=B` (A less/greater than B not making any difference) and if your checkouts file is `checkouts.cvs:A`, the following actions should be performed:

1. # **`mv checkouts.cvs:A checkouts.cvs:B`** (This provides the subsequent step with the appropriate checkouts file)
2. write a `supfile` whose collection line reads:

   ```
   src-all tag=B
   ```

3. cvsup your sources using the new `supfile`.

The `cvsup` utility will look for `checkouts.cvs:B`—in that the target is B; that is, `cvsup` will make use of the information contained therein to correctly manage your sources.

The benefits:

- the sources are dealt with correctly (in particular, no stale files)
- less load is placed on the server, in that `cvsup` operates in the most efficient way.

For example, `A=RELENG_4`, `B=.`. The period in `B=.` means -CURRENT. This is a rather typical update, from 4-STABLE to -CURRENT. While it is straightforward to "downgrade" your sources (e.g., from -CURRENT to -STABLE), downgrading a system is quite another matter. You are STRONGLY advised not to attempt such an operation, unless you know exactly what you are doing.

4.2 Updating to the same tag as of a different date

If you wish to switch from `tag=A` to `tag=A` as of a different GMT date (say, `date=D`), you will execute the following:

1. write a `supfile` whose collection line reads:

   ```
   src-all tag=A date=D
   ```

2. update your sources using the new `supfile`

Whether the new date precedes that of the last sync operation with `tag=A` or not, it is immaterial. For example, in order to specify the date "August 27, 2000, 10:00:00 GMT" you write the line:

```
src-all tag=RELENG_4 date=2000.08.27.10.00.00
```

> **Note:** The format of a date is rigid. You have to specify all the components of the date: century ("20", i.e., the 21st century, must be supplied whereas "19", the past century, can be omitted), year, month, day, hour, minutes, seconds—as shown in the above example. For more information, please see cvsup(1).

Whether or not a date is specified, the checkouts file is called `checkouts.cvs:A` (e.g., `checkouts.cvs:RELENG_4`). As a result, no particular action is needed in order to revert to the previous state: you have to modify the date in the `supfile`, and run `csvup` again.

4.3 Updating your ports collection for the first time

If you want to "sync" your ports tree for the first time you should use a tag that matches your FreeBSD installation. E.g., if you have installed the Ports Collection during the installation of FreeBSD 5.3-RELEASE the following should be used:

```
ports-all tag=RELEASE_5_3_0
```

The `cvsup` program will create a ports checkout file, which is precisely the goal of the first special sync operation. Now we can use the "." tag to update the tree whenever we feel it should be updated:

```
ports-all tag=.
```

All subsequent updates will be carried out smoothly.

If you have been reading the apparently nit-picking remarks in these sections, you will probably have recognized the potential for trouble in a source updating process. A number of people have actually run into problems. You have been warned. :-)