*Snap*Source: Automatic Snapshots of Source Codes

Hans Dulimarta
`http://www.egr.msu.edu/~dulimart`
`dulimart@computer.org`

Contents

WARNING: If you installed *Snap*Source version 0.10 or earlier, you have to update your configuration file by replacing comment character from pound ('#') to double slash (//).

1 Introduction

Right after Release 0.11 of *Snap*Source was announced, I got a comment in Freshmeat (http://freshmeat.net) from a user and he thought that *Snap*Source is YACVS (Yet Another CVS). *No, it is not!* In fact, if you read on you'll find out in Section 8 that I'm planning to synchronize information from the CVS local directory with that of *Snap*Source.

*Snap*Source was initially created for fulfilling my own need of some automatic development tools. In one of the projects I worked on, I had to try a number of different solutions to a problem before coming up with a final one. In experimenting with one solution, many changes had to be made to a set of source codes. Keeping the historical records of these changes across many different files in more than one subdirectory was a very tedious task.

It's true that RCS (Revision Control System) and Concurrent Version Control (CVS) provide facilities for managing modifications of our code. New versions are usually checked in when a "stable" release of the code is established. Checking in modifications which are not thoroughly tested will clutter the repository with unusable releases. Between two CVS commits, *Snap*Source will come in handy to provide you with a number of *micro* revision of your source code.

In our software projects, we often found ourselves amidst of modify-compile-test loops. The modification made to our codes can be either for fixing bugs or adding new features. During modification of the codes, it is possible that a new bug crept into our code. I thought that it would be nice to have an automatic tool which can periodically take a snap shot of our source codes, to facilitate browsing the historical records of what changes made to the codes as well as enabling us to revert the codes to an earlier version.

So, let there be ***Snap*Source** ...

2 What is *Snap*Source?

In short, *Snap*Source is a utility written in Perl to solve the problem mentioned above.

Creating the snapshot by merely *copying* the files at regular time interval will take a lot of space. Consider a project which has 100 Kbytes of source code. Taking the snapshot at an interval of 5 minutes will require a space of approximately 1.2 Mbytes after 1 hour. Usually changes made to the source codes between one snap shot to the next in a short time interval is very little, as little as few lines of codes. To overcome this space problem, the snap shot should store only the differences. *Snap*Source takes the later approach.

When *Snap*Source runs for the first time, it will recurse the source directory mentioned in the configuration files and then create a bunch of base files of the snapshot. Again, to conserve space, the base files are stored in compressed format. Currently, gzipped files are used. The next times *Snap*Source is invoked, it will create the difference (by running diff) between the current version of the source code and the corresponding base file. This difference file is referred to as delta file. The base files and delta files are stored in separate directories.

3 Supported Features

- *Snap*Source can recurse directory of more than one software project. Each project can be configured separately via a configuration file.
- Inclusion and exclusion of directory or file names can be controlled using three options `INCLUDE_PATTERS`, `EXCLUDE_PATTERS`, and `EXCLUDE_DIRS`.
- Automatic snapshots were taken by using `cron` facility. Activation of `cron` can be done by passing option `-i`.
- Differences between two snapshots can be displayed side by side (two-column format). This can be used for displaying the differences between a base and a delta file or between two delta files. Modified, changed, and deleted lines in the source code are shown in different colors for easy browsing.
- List all delta files of a selected source file.
- Search delta files by date. In most software development stage, modifications to source codes usually possess temporal locality, i.e. changes to a number of files made at about the same time. This feature might be useful in finding out this group of files, because usually changes made in this group are somewhat interrelated. The date specification must follow one of the following formats:
  1. Year only, such as in `2000` to search for all delta files created in year 2000, or
  2. Year and month, such as in `2000/9` to search for all delta files created in September 2000,
  3. A complete year, month, and date, such as in `2000/9/17` to search for all delta files created on that particular date.

  The search results can be selected and compared to its base file.
- Recalculate the new set of delta files when the base file of a particular source code is set to a newer revision date.
- Uninstallation of a specific project by specifying the `-u` option
- In the project tree display, directory names are tagged with folder icon to facilitate easy distinction from regular files.
- Display of detail information of each delta file: number of changes, deletions, and additions.

4 Obtaining *Snap*Source

If you have access to this file, I assume that you have downloaded the complete *Snap*Source tar file in one way of another. If not, you can obtain the tar file from the *Snap*Source's home page hosted at SourceForge (`http://sourceforge.net/projects/snapsource`). The current version of *Snap*Source is 0.17 .

5 Installation and Running

1. Put `snapsource` and `snapview` in a one of the directories in your PATH. If necessary, change the permission bit of both files with `chmod u+x`. Put `folder.xpm` in the same directory as `snapview`.
2. This utility has been tested using Perl/Tk version 800.018. If the `widget` demo script of this package runs, you should be able to run *Snap*Source. Otherwise, you might have to set your PERL5LIB to the path where Perl/Tk is installed in your system.
3. Create directory `.snapsource` under your home directory
4. Edit your *Snap*Source configuration file and copy it to the `.snapsource` directory created in the previous step.

```
cp path-to-config-file/MyProject home-directory/.snapsource
```

*Snap*Source consists of two utilities `snapsource` and `snapview`.

5.1 Running `snapsource`

- To run *Snap*Source manually:

```
snapsource MyProject
```

- To run *Snap*Source from `cron`, use the "install" option:

```
snapsource -i MyProject
```

  This is the recommended way of running `snapsource`.

- To stop running *Snap*Source on a specific project, use the "uninstall" option:

```
snapsource -u MyProject
```

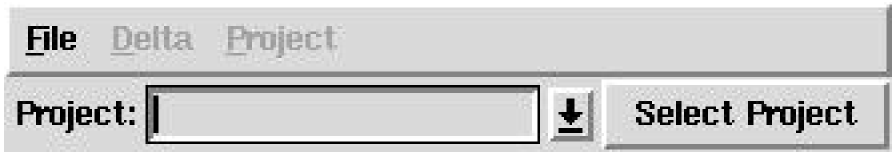


Figure 1: Main window of `snapview`.

5.2 Running `snapview`

The utility `snapview` is the GUI for browsing through all the delta files created by `snap-source`. The first screen that you will see after running `snapview` is shown in Figure 1.

After this point, you can select a project by clicking the down arrow on the pull-down menu to bring up a list of all project configuration files found in the `~/.snapsource` directory. After the project name is chosen and the "Select Project" button is clicked, a tree display of the selected project will show up (Figure 2). In the current implementation, the tree display does not distinguish between ordinary file and directory. However, if a directory name on the tree display is double clicked, the content of that directory will be shown and the tree display will expanded accordingly. If an ordinary file is double clicked, the delta compare window in Figure 3 will show up.

Comparing files maintained by *Snap*Source can be done in two ways:

- Comparing a base file and a delta file
- Comparing two delta files with different date of snapshot

This choice can be selected by clicking the radio button on top of the window shown in Figure 3. When "Base & Delta" is selected, only one date box will be visible, but when "Delta & Delta" is selected, two date boxes will be visible for selecting the snapshot date of the first and second delta files.

The output of comparing delta files (base–delta or delta–delta) will be similar to that shown in Figure 4.

5.3 Searching for Delta Files

In an active software development project, its source code will continually changing, and hence a large number of delta files will be produced when *Snap*Source is used to maintain the project. To facilitate easy selection of delta files, *Snap*Source provides a "Search by Date" facility which can be invoked from the DELTA::SEARCH BY DATE menu. As a result, a window in Figure 5 will show up.

The "Revision Date" entry field can be given in three formats: `YYYY/MM/DD`, `YYYY/MM`, or `YYYY`. The most complete format will search for delta files which are produced on a specified date. The least complete format will search for delta files in a specified year.

To speed up changing the date entry, a combination of radio button and push button can be used. The PREV and NEXT will change the date to a less and more recent date, respectively. Figure 6 shows a typical search result.

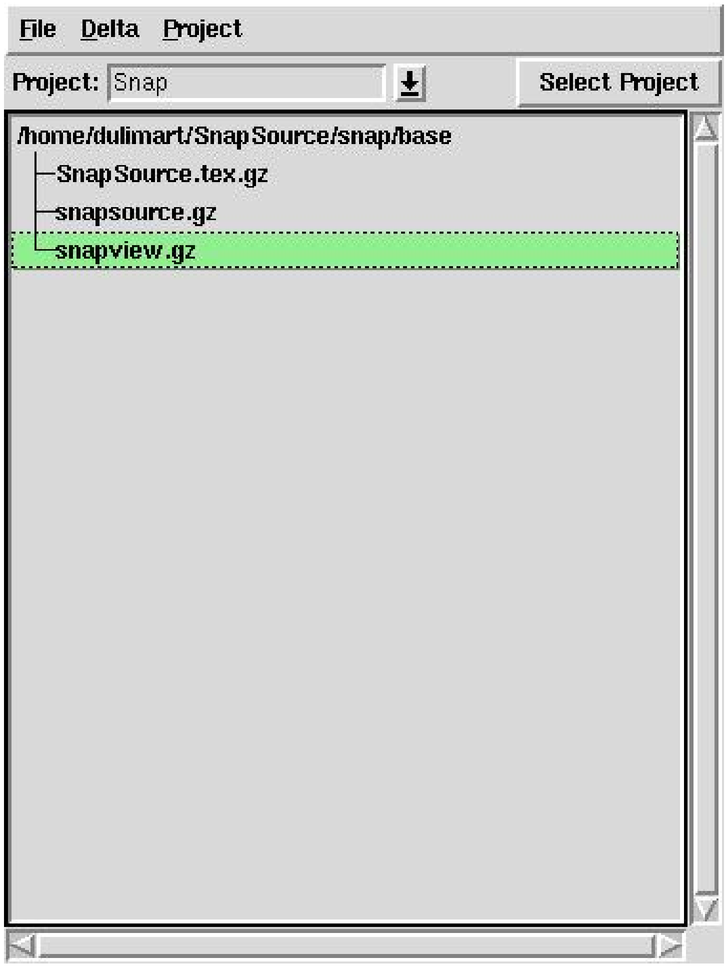


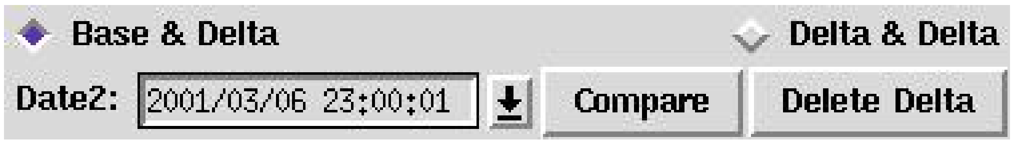
Figure 2: Tree display of a project.

Another facility provided by *Snap*Source is listing of all delta files. When the DELTA::LIST ALL menu is selected, a window similar to Figure 7 will be shown. Two commands are available on the window: REASSIGN BASE and DELETE DELTA. Both commands work on the selected file from the list. DELETING DELTA is self-explanatory. Further detail description will be given to the REASSIGN BASE command.

For each source file managed by *Snap*Source, a compressed base file will be created. On each periodic snapshot, a delta file will be created for the source whenever the current content of the source differs from the uncompressed base file. All subsequent delta files will be created based on the difference to this base file.

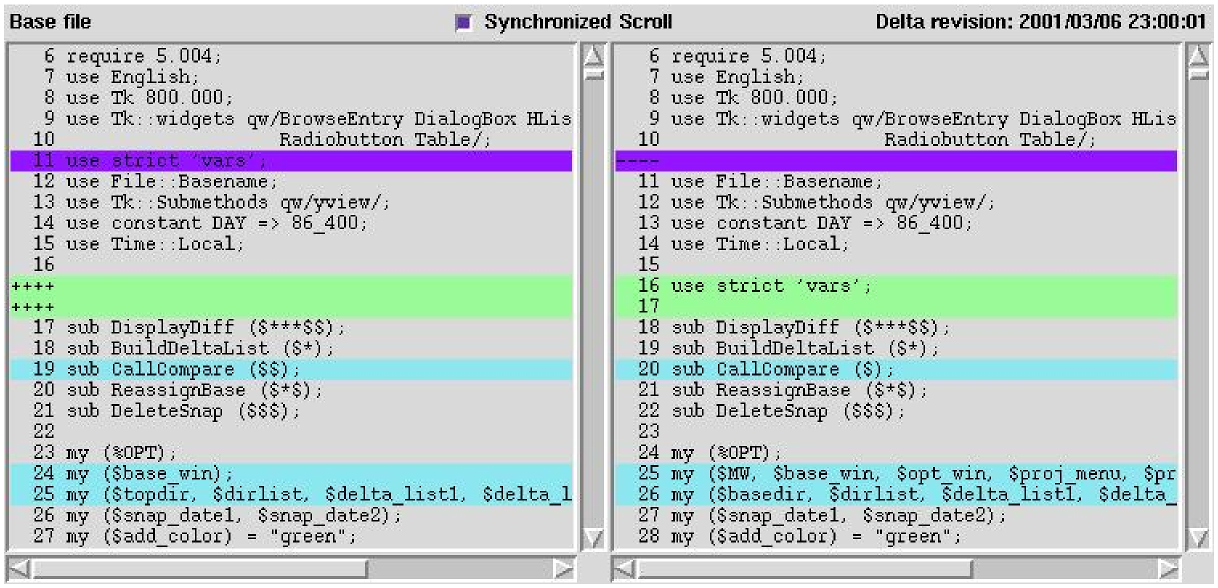
When a base file is reassigned to a new revision date three things have to be carried out:

1. A new compressed base file has to be created
2. All delta prior to the given revision date has to be deleted
3. All delta after the given revision date has to be updated



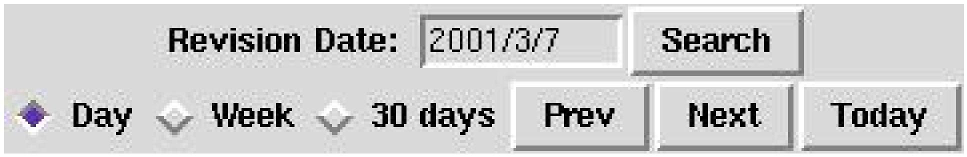
[✓ Base & Delta] Delta & Delta
Date2: 2001/03/06 23:00:01

Figure 3: Comparing base and delta file of a project.


Base file | [✓ Synchronized Scroll] | Delta revision: 2001/03/06 23:00:01

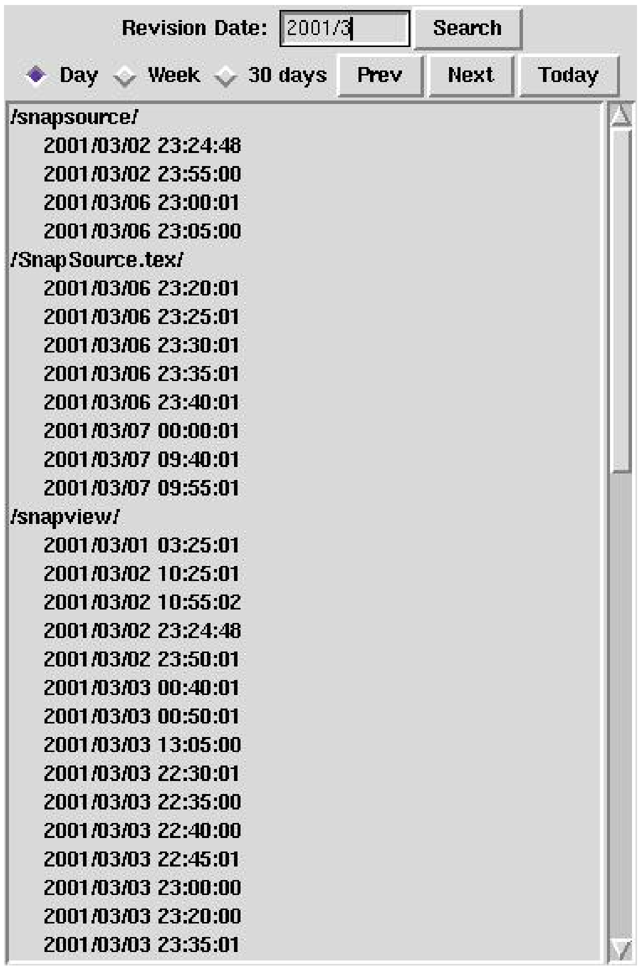
```
  6 require 5.004;                                 6 require 5.004;
  7 use English;                                   7 use English;
  8 use Tk 800.000;                                8 use Tk 800.000;
  9 use Tk::widgets qw/BrowseEntry DialogBox HLis  9 use Tk::widgets qw/BrowseEntry DialogBox HLis
 10                   Radiobutton Table/;         10                   Radiobutton Table/;
 11 use strict 'vars';                           ----
 12 use File::Basename;                           11 use File::Basename;
 13 use Tk::Submethods qw/yview/;                 12 use Tk::Submethods qw/yview/;
 14 use constant DAY => 86_400;                   13 use constant DAY => 86_400;
 15 use Time::Local;                              14 use Time::Local;
 16                                               15
++++                                              16 use strict 'vars';
++++                                              17
 17 sub DisplayDiff ($***$$);                     18 sub DisplayDiff ($***$$);
 18 sub BuildDeltaList ($*);                      19 sub BuildDeltaList ($*);
 19 sub CallCompare ($$);                         20 sub CallCompare ($);
 20 sub ReassignBase ($*$);                       21 sub ReassignBase ($*$);
 21 sub DeleteSnap ($$$);                         22 sub DeleteSnap ($$$);
 22                                               23
 23 my (%OPT);                                    24 my (%OPT);
 24 my ($base_win);                               25 my ($MW, $base_win, $opt_win, $proj_menu, $pr
 25 my ($topdir, $dirlist, $delta_list1, $delta_l 26 my ($basedir, $dirlist, $delta_list1, $delta_
 26 my ($snap_date1, $snap_date2);                27 my ($snap_date1, $snap_date2);
 27 my ($add_color) = "green";                    28 my ($add_color) = "green";
```

Figure 4: Two-column output of comparing base and delta file



Revision Date: 2001/3/7
[✓ Day] Week 30 days

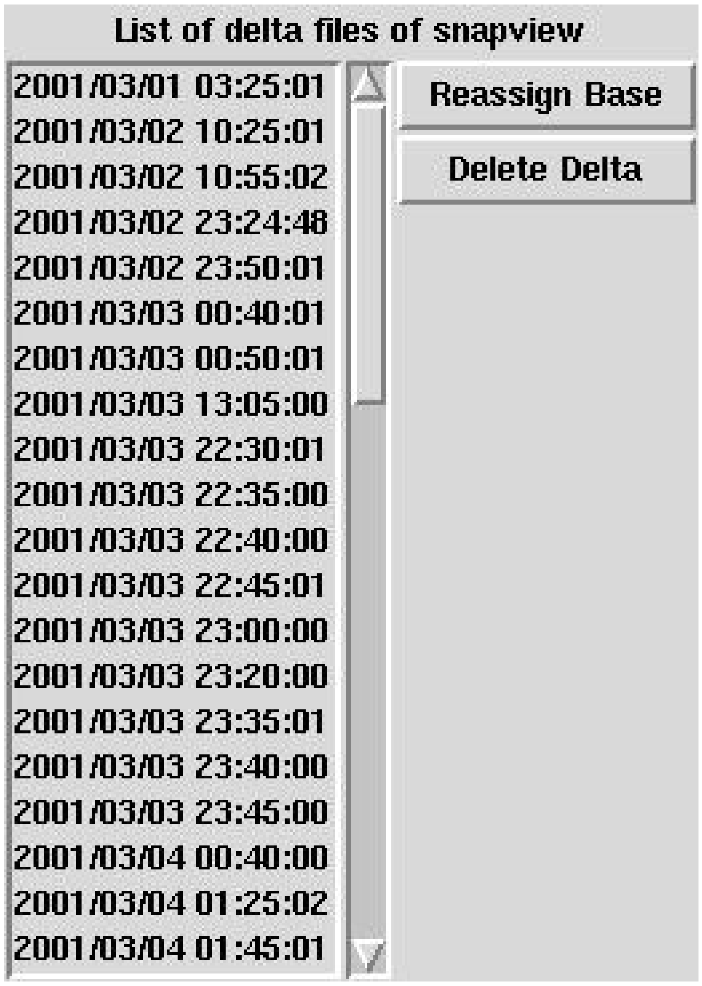
Figure 5: Searching delta files by date



```
/snapsource/
    2001/03/02 23:24:48
    2001/03/02 23:55:00
    2001/03/06 23:00:01
    2001/03/06 23:05:00
/SnapSource.tex/
    2001/03/06 23:20:01
    2001/03/06 23:25:01
    2001/03/06 23:30:01
    2001/03/06 23:35:01
    2001/03/06 23:40:01
    2001/03/07 00:00:01
    2001/03/07 09:40:01
    2001/03/07 09:55:01
/snapview/
    2001/03/01 03:25:01
    2001/03/02 10:25:01
    2001/03/02 10:55:02
    2001/03/02 23:24:48
    2001/03/02 23:50:01
    2001/03/03 00:40:01
    2001/03/03 00:50:01
    2001/03/03 13:05:00
    2001/03/03 22:30:01
    2001/03/03 22:35:00
    2001/03/03 22:40:00
    2001/03/03 22:45:01
    2001/03/03 23:00:00
    2001/03/03 23:20:00
    2001/03/03 23:35:01
```

Revision Date: 2001/3 [✓ Day] Week 30 days

Figure 6: Search Result



```
2001/03/01 03:25:01
2001/03/02 10:25:01
2001/03/02 10:55:02
2001/03/02 23:24:48
2001/03/02 23:50:01
2001/03/03 00:40:01
2001/03/03 00:50:01
2001/03/03 13:05:00
2001/03/03 22:30:01
2001/03/03 22:35:00
2001/03/03 22:40:00
2001/03/03 22:45:01
2001/03/03 23:00:00
2001/03/03 23:20:00
2001/03/03 23:35:01
2001/03/03 23:40:00
2001/03/03 23:45:00
2001/03/04 00:40:00
2001/03/04 01:25:02
2001/03/04 01:45:01
```

Figure 7: Listing all delta files

This process will be describe formally as follows. Suppose a compressed base file B and n delta files: $D_1, D_2, \ldots, D_n$ are currently maintained, and the base file will be assigned to the revision date of D_j's.

1. B_u = uncompress (B)
2. B' = patch (B_u, D_j).
3. Delete $D_1, D_2, \ldots, D_j$
4. For $k = j + 1, \ldots n$: D_k = diff (B', patch (B_u, D_k))
5. B = compress (B')

6 Configuration File

In order to create the snapshots, *Snap*Source has to be informed where to check the source codes and where to store the base and delta files. A configuration file is read from the `.snapsource` directory in your home directory. Lines beginning with a double slash sign (`//`) in the configuration file are considered as comment lines and will be ignored by `snapsource`. **WARNING:** Starting in release 0.11, the old comment delimiter ('#') was obsoleted, because the pound sign was used by Tk to represent color names.

In general, a *Snap*Source configuration file is line-oriented and each line consists of a parameter name and its value delimited by an equal sign. All configuration entries have to be written not to exceed a single line.

6.1 Interval

To perform automatic and periodic capture of source code, *Snap*Source must be triggered by some external event. The `INTERVAL` parameter specifies how often the external event should be generated. The value specified is given in minutes.

6.2 Source and Target Directory

In the configuration file, `SOURCE_DIRECTORY` specifies the top-level directory where the source code of the project is stored. This directory will be recursively checked by *Snap*Source. The `TARGET_DIRECTORY` specifies the directory to store the base and delta files. Under this target directory, *Snap*Source will create two subdirectories: `base` (for storing the base files) and `delta` (for storing the delta files).

The structure of `SOURCE_DIRECTORY` will be mimicked in the target directory as `TARGET_DIRECTORY/base` and `TARGET_DIRECTORY/delta`. Files in the delta directory will have a suffix

`diff.`*YYYYMMDDhhmmss*

where *YYYY* is the year, *MM* is the month (01–12), *DD* is the day (01–31), *hh* is the hour (01–23), *mm* is the minute (00–59), and *ss* is the second (00–59) when the difference was created.

6.3 File Inclusion or Exclusion

With thousands of programming language out there in the software world, *Snap*Source has to anticipate the possibility that the source codes it has to take care of can be written in anyone of these languages. Therefore, users are given choice to selectively include or exclude file names by some patterns. The parameters `INCLUDE_PATTERNS` and `EXCLUDE_PATTERNS` are provided for this purpose. Patterns can be given as if they were shell wild cards.

Starting in Revision 0.15, *Snap*Source offers an option for excluding directory names as well. This behavior is controlled by the parameter `EXCLUDE_DIRS`. Any directories whose name matches any of the patterns specified in this option will be excluded from the search by `snapsource`. This also implies that all the files and lower subdirectories under those directories will be excluded.

6.4 Colors

To facilitate easy visual location of changes in the source code, *Snap*Source displays them in different colors. To accommodate different window manager themes and color settings, user preferred colors can be specified in the configuration file.

The parameters `ADD_COLOR`, `CHANGE_COLOR`, and `DELETE_COLOR` can be set to any color names in the X windows color name data base or a color constant prefixed with a pound sign ('#') as used by `Tk_Color`.

6.5 Options

*Snap*Source is designed to incorporate extensive user customization preferences. There are a number of different spots in which user preferences can be incorporated in to the *Snap*Source utility: compression method for creating base files, diff command options for creating delta files, delta file suffix format, etc.

In this release of *Snap*Source, the parameter `DIFF_OPTION` can be used to specify how delta files are to be generated.

6.6 Running `crontab`

Starting release 0.10, a new option for specifying location of the `crontab` program is available. If this variable is undefined, `CRONTAB` will be set to `/usr/bin/crontab`.

7 Requirements

Perl/Tk has to be installed for the `snapview` to run properly.

8 Planned Additional Features

- Command for reverting source code to a particular previous revision date
- Multiple selection on DELETE DELTA command.
- Use vixie cron features, if it is installed.
- Incorporate information from the CVS or RCS directory.
- Modularized Perl code

9 Target and Source Directory Coherence

Some questions that need to be addressed:

- What to do if source files are moved to another directory?
- What to do if source files are renamed?

10 Bugs Report or Feature Requests

If you find any unexpected behavior of *Snap*Source or have some suggestions for improvement, please contact me by e-mail at `<dulimart at computer.org>`.

When you report a bug, please include the following:

- Output of `snapsource -v YourConf`, where `YourConf` is your *Snap*Source configuration file
- A copy of `YourConf`
- Description of the bug and how to reproduce it
- E-mail output generated by `crontab`, if you installed `snapsource` as a cron job.