Table of Contents

Personal Development Branches..1

Personal Development Branches

In the case you are using the trunk only you do not have revision control for the small changes you are doing. You shall not commit non-debugged or non-compiling code to the trunk. However you might want to commit the first compiling version before debugging so you can track the changes. If you copy the trunk to your own working branch you can do this. So you check out a working copy of your private branch and a working copy of the trunk.

1. Create your own branch by copying the trunk:

   ```
    svn cp --username <username> svn://pi.physik.uni-bonn.de/MarlinTPC/trunk svn://pi.physik.
   ```
2. Check out a working copy of your branch:

   ```
   svn checkout svn://pi.physik.uni-bonn.de/MarlinTPC/branches/username MarlinTPC_working_bra
   ```
3. Make your changes. You may commit as often as you like to your personal development branch and use all the other features of version control. Since this is your private branch, there should be no conflicts when committing. (However, someone else could check out your branch and make changes. But this is not how it is meant to be.)
4. If someone committed changes to the trunk you have to merge them into your branch. It is recommended that you commit all your changes before merging. Unfortunately subversion does not remember which data from the trunk has already been merged, but you can see it from your log (`svn log`). If you merge the trunk for the first time, it is the revision when you created your branch (step 1), after that it is the revision of the previous merge (see log entry step 5). If this revision for instance was 132:

   ```
    svn merge -r132:HEAD svn://pi.physik.uni-bonn.de/MarlinTPC/trunk
   ```

   This step merges the changes that were committed to the trunk into your working copy ouf your personal development branch. It behaves like step 3 in the description for using the trunk only, causing conflicts if automatic merging failed. These conflicts are indicated by the status character "C" in front of the affected file and have to be resolved by hand (for more details have a look at the subversion book, in particular the "Resolve Conflicts" section). After merging you should test and debug again. Remember which was the revision of the head (say, 174 in this example), you need it in the next step.
5. Commit the debugged, working software with the latest changes of the trunk in them to your private branch. Mention which revision was merged, so you can look it up for the next merge.

   ```
    svn commit -m "Merged r174 from trunk."
   ```
6. To commit changes to the trunk, you first have to check out a working copy of it:

   ```
   svn checkout svn://pi.physik.uni-bonn.de/MarlinTPC/trunk MarlinTPC_trunk
   ```

   In case you already have a working copy of the trunk, you should update it with `svn update` (performed in the working copy directory of the trunk).
7. Before you merge your changes into the trunk you must perform steps 4 and 5. BEWARE: If you omit step 4 and 5 you will undo changes already commited to the trunk. After these steps your personal development branch contains both your new developments and the recent changes of the trunk. Now you can merge your personal develepment branch into your working copy of the trunk (performed in the working copy directory of the trunk):

   ```
   svn merge svn://pi.physik.uni-bonn.de/MarlinTPC/trunk@<REV> svn://pi.physik.uni-bonn.de/Ma
   ```

   `<REV>` is the revision you want to merge, `<username>` is the name out your developer branch.
8. Make sure there are no conflicts and the trunk compiles!
9. Finally you commit the changes to the trunk (performed in the working copy directory of the trunk):

```
        svn commit -m "Log message of the changes I made"
```

When performing a write operation on the repository (like commits or creating your private branch), Subversion forces you to create a log message. There are three ways to provide this message:

- For short message you can include it in your svn command by adding `-m "Short log message"`
- For longer messages with several lines you can create a file and give this in the svn command: `-F file_with_log_message.txt`
- If none of the above is specified, svn opens an editor (according to the EDITOR environment variable) where you can enter your log message. When you save the file and quit the editor the svn command is performed.

This is not an introduction to all the subversion commands and topics like solving conflicts etc. To get a list of available commands type `svn help`. To get help about a specific command type `svn help command`. I recommend reading the first two chapters of the subversion book, dealing with fundamental concepts and basic usage. This takes one or two hours, but you will regain them in profiting from a very powerful, easy-to-use version control system.

-- MartinKillenberg - 26 Feb 2008

This topic: ILCTPC > DeveloperWorkbookDevelopmentBranches
Topic revision: r6 - 2009-06-24 - StefanoCaiazza