branches of a file into the user's working copy, thus allowing the changes to be committed to some other branch.

1.1 Creating Branches

A branch in CV


1.1
1.2
1.3
1.4
1.5

A rev s on number s an *even* number of ntegers separated by per ods. A branch

The below example would reassign the tag '`r1-0`' to the branch 1.4.2 (with magic branch number 1.4.0.2) for the file '`oo.c`'.

```
# cvs admin -Nr1-0:.4.2 oo.c
```

```
# cvs upd t  -j r l- -0
[...]
# cvs comm_t -m "M rg d c  ng s  rom r l- -0."
```

L fe s naturally not always as s mple as the above e_ample would n d cate. Qu te often the merg ng of changes w ll result n conf l cts. It s *absolutely imperative* to resolve any conf l cts that ar se from the update be fore comm tt ng the new vers on, lest the repos tory become muddled w th broken files full of random conf l cts.

3.1 Merging wi h Keywords

When sw tch ng rev s ons frequently t may become confus ng e_actly wh ch rev s on of a f n

In the re$s tory all keywords n files are stored une_panded, e n the r or g nal form. W

4.1 Initializing the Repository

The first step necessary is to create a repository to play with. The most convenient place to create one is in the $HOME directory. So, logged in as a non root[1] user we create a repository in our home directory.

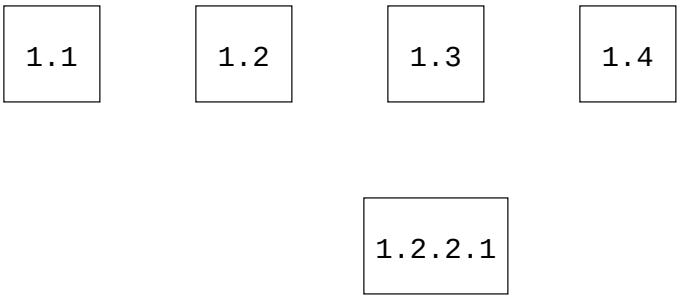
```
# cd
# mkdir cvs
# export CVSROOT=$HOME/cvs
# cvs init
# ls cvs
CVSROOT
```

Following this we need a place to play. So we create a sandbox that will contain our working copy. Once we have a sandbox we need to associate a project and commit it to our repository. Usually this is done by associating a directory, using '`cvs import`' on the directory, and then removing the imported directory and checking it out from the repository. This is a bit laborious for a simple toy, so we will cheat a little by manually associating an empty directory in the repository, then checking it out.

```
# mkdir sandbox
# mkdir cvs/foo
# cd sandbox
# cvs checkout foo
cvs checkout: Updating foo
```

Now we have a project that we can actually play with. We need to

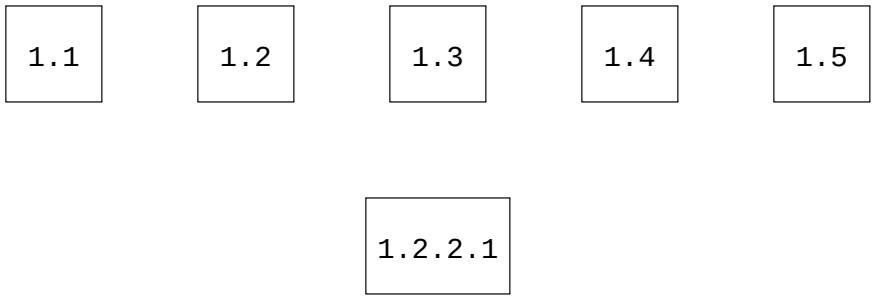
We have created a revision on this branch and we now find the revision number is what we originally expected. Revision 1.2.2.1 is not the zeroth, but the first revision of this file on branch 1.2.2.


1.1
1.2
1.3
1.4
1.2.2.1

```
# c t  oo
 oo
```

```
   Sticky Tag:           (non
[...]
```

Since the files in your working copy were checked out from the main trunk that is where they were committed to, and the repository now contains a new revision, 1.5. The updated revision tree is shown in Figure 8, depicting the merge of 1.4 and 1.2.2.1 together to form 1.5.



```
   - ,3 + ,2
  oo
-b r
- nord
+ oob r
```

The preced ng d ff

```
corg   l rp
>>>>>>>  .2.2.3
```

The l ne ωonta n ng the str ng “foobar” s the ωulpr t. That part ωular str ng was absorbed nto the ma n trunk when we merged rev s on

5 Conclusion

At first encounter, CVS branches seem confusing. They become perhaps more confusing when we realize that they are implemented using tags. But branches in CVS are not as conf