```
$ svn help
```

Importing Files and Directories

```
$ svnadmin create http://svn.example.com/svn/repo/some/project \
```

Importing Files and Directories

```
$ svn import /path/to/mytree \
http://svn.example.com/svn/repo/some/project \
-m "Initial import"
Adding mytree/foo.c
Adding mytree/bar.c
Adding mytree/subdir
Adding mytree/subdir/quux.h
Committed revision 1.
$
$ svn list http://svn.example.com/svn/repo/some/project
bar.c
foo.c
subdir/
$
```

Creating a Working Copy

```
$ svn checkout http://svn.example.com/svn/repo/trunk
A trunk/README
A trunk/INSTALL
A trunk/src/main.c
A trunk/src/header.h
```

Basic Work Cycle

1. *Update your working copy.*
svn update

2. *Make your changes.*
svn add, **svn delete**, **svn copy**, **svn move** and **svn mkdir**

3. *Review your changes.*
svn status and **svn diff**

4. *Fix your mistakes.*
svn revert

5. *Resolve any conflicts (merge others' changes).*
In the time it takes you to make and review your changes, others might have made and published changes, too. You'll want to integrate their changes into your working copy to avoid the potential out-of-dateness scenarios when you attempt to publish your own. Again, the **svn update** command is the way to do this.
If this results in local conflicts, you'll need to resolve those using the **svn resolve** command.

6. *Publish (commit) your changes.*
svn commit

NOTE:

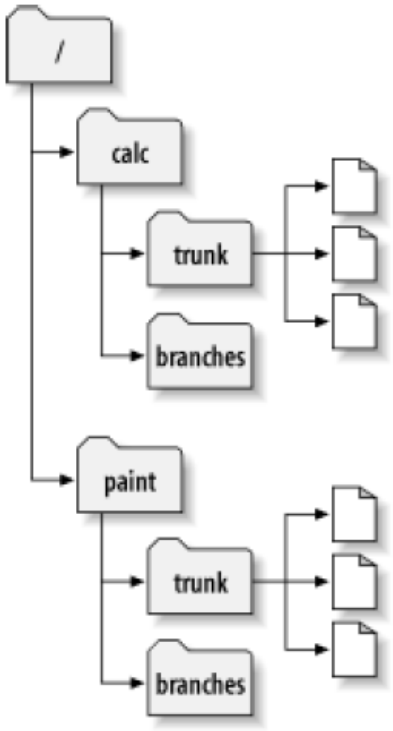
```
$ svn status stuff/fish.c
```

```
$ svn status -v
M      44    23    sally       README
       44    30    sally       INSTALL
M      44    20    harry       bar.c
       44    18    ira         stuff
       44    35    harry       stuff/trout.c
D      44    19    ira         stuff/fish.c
       44    21    sally       stuff/things
A      0     ?     ?           stuff/things/bloo.h
       44    36    harry       stuff/things/gloo.c

$ svn status -u -v
M  *   44    23    sally       README
M      44    20    harry       bar.c
   *   44    35    harry       stuff/trout.c
D      44    19    ira   stuff/fish.c
A      0     ?     ?     stuff/things/bloo.h
Status against revision: 46

$ svn revert sandwich.txt
```

Branching and Merging



Commit changes bit by bit will surely break things for others.

Not commit or update until your work is done.

There are a number of problems with this, though.

First, it's not very safe. Should something bad happen to your working copy or computer, you risk losing all your changes.

Second, it's not very flexible. Replicate your changes across different computers. You cannot share your work with anyone else.

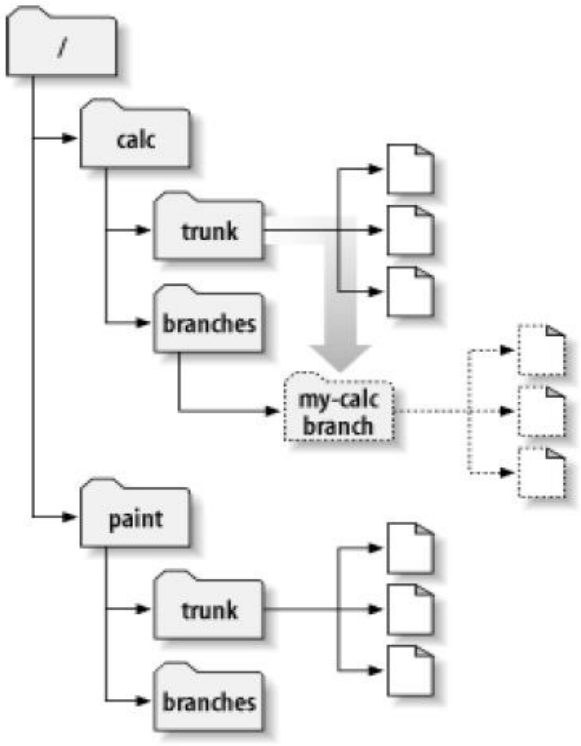
Finally, when you're finished with all your changes, you might find it very difficult to merge your completed work with the rest of the company's main body of code.

The better solution is to create your own branch, or line of development, in the repository. This allows you to save your not-yet-completed work frequently without interfering with others' changes and while still selectively sharing information with your collaborators. You'll see exactly how this works as we continue.

You'll create a new directory, `/calc/branches/my-calc-branch`, which begins its life as a copy of `/calc/trunk`.

```
$ svn copy http://svn.example.com/repos/calc/trunk \
http://svn.example.com/repos/calc/branches/my-calc-branch \
-m "Creating a private branch of /calc/trunk."
Committed revision 341.
$
```

Create the branch on the server side!!!



```
$ svn checkout http://svn.example.com/repos/calc/branches/my-calc-branch
A my-calc-branch/Makefile
A my-calc-branch/integer.c
A my-calc-branch/button.c
Checked out revision 341.
$
```

Notice that Subversion is tracing the history of your branch's `integer.c` all the way back through time, even traversing the point where it was copied.

Basic Merging

The good news is that you and Sally aren't interfering with each other. The bad news is that it's very easy to drift *too* far apart and it may be near-impossible to merge your changes back into the trunk without a huge number of conflicts.

Instead, you and Sally might continue to share changes as you work. It's up to you to decide which changes are worth sharing; Subversion gives you the ability to selectively "copy" changes between branches. And when you're completely finished with your branch, your entire set of branch changes can be copied back into the trunk. In Subversion terminology, the general act of replicating changes from one branch to another is called *merging*, and it is performed using various invocations of the **svn merge** subcommand.

Continuing with our running example, let's suppose that a week has passed since you started working on your private branch. Your new feature isn't finished yet, but at the same time you know that other people on your team continue to make important changes in the project's `/trunk`. It's in your best interest to replicate those changes to your own branch, just to make sure they mesh well with your changes. This is done by performing a *sync merge*—a merge operation designed to bring your branch up to date with any changes made to its ancestral parent branch since your branch was created.

Frequently keeping your branch in sync with the main development line helps prevent "surprise" conflicts when the time comes for you to fold your changes back into the trunk.

Subversion is aware of the history of your branch and knows when it split away from the trunk. To perform a sync merge, first make sure your working copy of the branch is "clean"—that it has no local modifications reported by **svn status**. Then simply run:

```
$ pwd
/home/user/my-calc-branch
$ svn merge ^/calc/trunk
--- Merging r345 through r356 into '.':
U button.c
U integer.c
--- Recording mergeinfo for merge of r345 through r356 into '.':
U .
$
```

This basic syntax—**`svn merge` *`URL`***—tells Subversion to merge all changes which have not been previously merged from the URL to the current working directory (which is typically the root of your working copy). Notice that we're using the caret (`^`) syntax[2] to avoid having to type out the entire `/trunk` URL. Also note the "Recording mergeinfo for merge…" notification. This tells you that the merge is updating the `svn:mergeinfo` property.

After running the prior example, your branch working copy now contains new local modifications, and these edits are duplications of all of the changes that have happened on the trunk since you first created your branch:

```
$ svn status
M .
M button.c
M integer.c
$

$ svn commit -m "Merged latest trunk changes to my-calc-branch."
Sending .
Sending button.c
Sending integer.c
Transmitting file data ..
Committed revision 357.
$
```

At this point, your private branch is now “in sync” with the trunk, so you can rest easier knowing that as you continue to work in isolation, you're not drifting too far away from what everyone else is doing.

Memo

```
svn co svn+ssh://minshan@hyperspec-nas.ee.e.uh.edu/volume1/svn/test
/home/minshan/test_svn


!svnadmin create /data/mysvn_repo
!svn import /data/tookit file:///data/mysvn_repo -m "Initial import"
!svnlook tree /data/mysvn_repo/
!svn co file:///data/mysvn_repo co_repo

!svn status

!svn log
!svn log file
!svn log --limit 2
!svn log -r 2
!svn log -r 4:6(6:4)
!svn log -v
!svn log -v -r 3

!svn diff file (only used for text files)
!svn diff -r 1:3 file


!vim flda.m
:x %exit at the same time
:q! %exit without saving
:wq

!svn update
!svn ci -m "put the log file."
!svn update

!svn add file.txt
!svn delete(rm) file
!svn copy ori_file new_file
!svn mkdir file

!svn revert file(.txt) (modify add delete)


NOTE: Making changes to the file can be done by direct editing the
file but adding a new file is not permitted. It will show ? mark which
means this file is not under the subversion control.
```