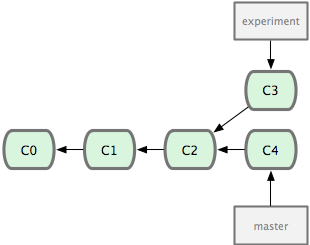
**Linear Branching Pattern:**

1. In Linear Branching pattern , we get cleaner project history.
2. The changesets are arranged in linear fashion, irrespective of the commits done in branches other than the master.
3. We can directly access the files at particular version, since it avoids 3 way merging.
4. We can implement this linear branching pattern in git, using **git rebase** command.
5. LBP can be implemented in CVS by connecting the 2 branches with their latest revision numbers.

**Examples:**

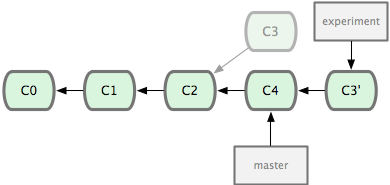
**LBP accomplished through git:**

Let us consider that c3 is a commit done on experiment branch and c4 is a commit done on master branch.The parent commit is c2 .

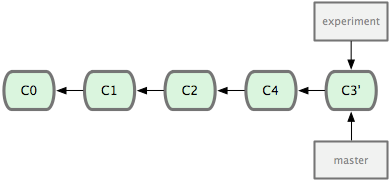


In order to merge these c3 and c4 commits,git rebase can be used.

Git rebase creates additional commit called c3’,Which vanishes c3 commit ,by adding the difference between c2 and c3 to the commit c2.Now c2 has c3 changes which is in link with c4 thorugh c3’



Now the commit history is clean and it is in linear fashion. Branches are merged.



**LBP accomplished in CVS:**

Here in cvs.the revision numbers are same as that of commit in git.The revision numbers which represents the files history are in linear fashion.

+-----+ +-----+ +-----+ +-----+ +-----+

! 1.1 !----! 1.2 !----! 1.3 !----! 1.4 !----! 1.5 !

+-----+ +-----+ +-----+ +-----+ +-----+

To the main trunk/revision line a branch can be created wherever needed as shown below (ie) 1.2.2.1

+---------+   
mynew

Branch -> \_\_! 1.2.2.1 !\_\_\_\_ <- the maintenance   
 / +---------+ \   
 / \  
 / \  
 +-----+/ +-----+ +-----+ \+-----+  
 ! 1.0 !---! 1.1 !----! 1.2 !----! 2.0 ! <- The main trunk  
 +-----+ +-----+ +-----+ +-----+

Now the changes can be merged to main trunk by switching to 1.4 (latest revision) and merging with 1.2.2.2 revision of newly created branch.

**2) Continuous Integration Branching pattern**

Facilitates parallel development among team members.  
This pattern are used when the application is to be autodeployed.

Whenever changes are to be commited to the trunk in svn,we can automatically deploy from it.

Tow types in CI is possible. **Feature branching** -> CI done when feature branch is added to master/trunk(Svn)

**Task branching** -> CI done when the issue/task is completed in separate branch called task branch and merged to master.then automatically deployed and issue is fixed in application

Example:

Let us consider feature branch created from master has its functionality working fine.

Now It can be merged with master successfully.We can use CI branching pattern in a such a way that whenever merge commit is created on master branch,it has to be deployed automatically.After deployment this feature works fine in application.

