Name: Rouise Tan Yu Shuang (A4049)

1. Introduction

Version control software are a category of software tools which can help a software team or software student manage the changes to the source code over the time. Version control software always keep track of every modification to the code in other database. If make mistake, they can turn back the clock and compare earlier version of the code in order to help them solve the problems and minimizing the mistake in the code.

Version control software is provide at some of the sides such as Github. This site typically build a suit service around the version control. It allow to release downloads, mailing lists, bug trackers, web hosting and build farms. This range of functionality makes them particularly attractive for the projects that do not have the resources to maintain their own server for version control.

Good version control software supports a developer's workflow without stately on a particular way of working. Basically, it also works on any platforms, rather than dominate what operating system or which tool chain developers must use. Also, it facilitate smooth and continuous flow of changes to the code instead of a clumsy mechanism of file locking. Meaning that giving permission to a developer at the expense of blocking the progress of others. However, the software teams which do not use any form of version control software run into problems like not knowing which can make changes.

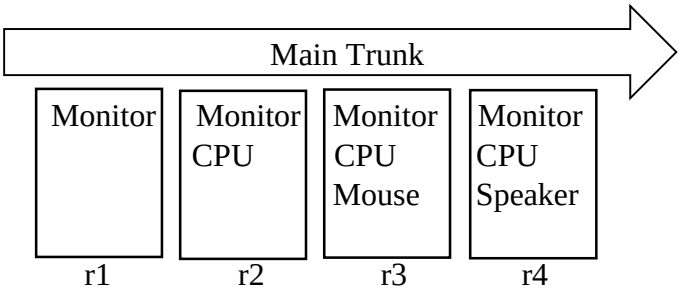
In software engineering, one of the most popular version control system in use for nowadays is called Git. Git is a distributed version control software. Also, Git is free and open source. Git is necessary to be used because many individual overs the year made changes and those changes include the creation and deletion of the files as well as modify their contents. Different version control tools differ on how they handle renaming and how to move the files. However, it enable return to the previous versions to help in root causes analysis for bugs and it is decisive when need to solve the problems in older versions of the software.

To discuss this version control software, some basic git commands (basic actions) and advances actions could state which are:

Basic git commands	Explanations	Source code
Add	Put a file into the repositories for the first time. For example begin tracking it with the version control software.	git add <filename>
Revision	What version a file is on (version1, version2, or version3.)	
Head	The latest revision in the repositories.	git show | head
Check out	Download a file from the repositories.	git checkout -- <filename>
Check in	Upload a file to the repository (if it has changed). The files gets a new revision number, and user can 'check out' the latest one.	git push
Checkin Message	A short message describing what was changed.	
Chagelog/History	A list of changes made to a file since it was created.	git log
Update/Sync	Synchronize your files with the latest from the repository. This lets you grab the latest revisions of all files.	git pull
Revert	Throw away your local changes and reload the latest version form the repository.	git revert

2. Checkin

Check in a file (list.txt) is the simplest scenario and it can modifying it over time.



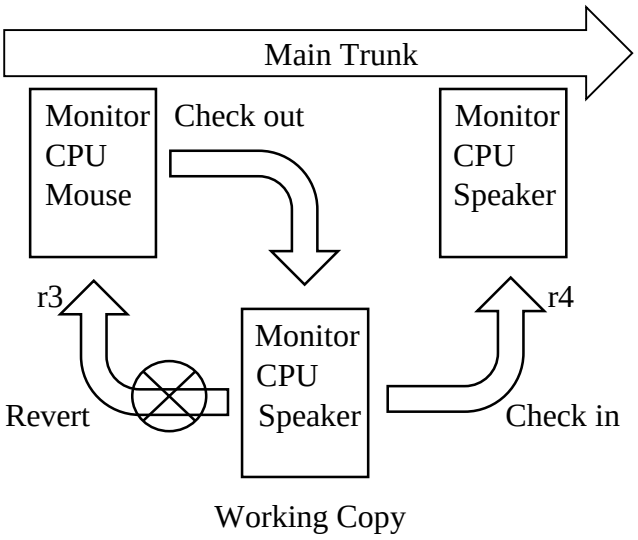
Each time user can check in a new version and get a new revision such as (revision1, revision2 and revision3). It can do as:

Svn add list.txt
(modify the file)
svn ci list.txt –m “Changed the list”

:: -m is the message to use for this checkin.

3. Checkouts and Editing

In reality, some of the user not keep checking in a file. Actually, users have to check out, edit and check in. The cycle looks as:



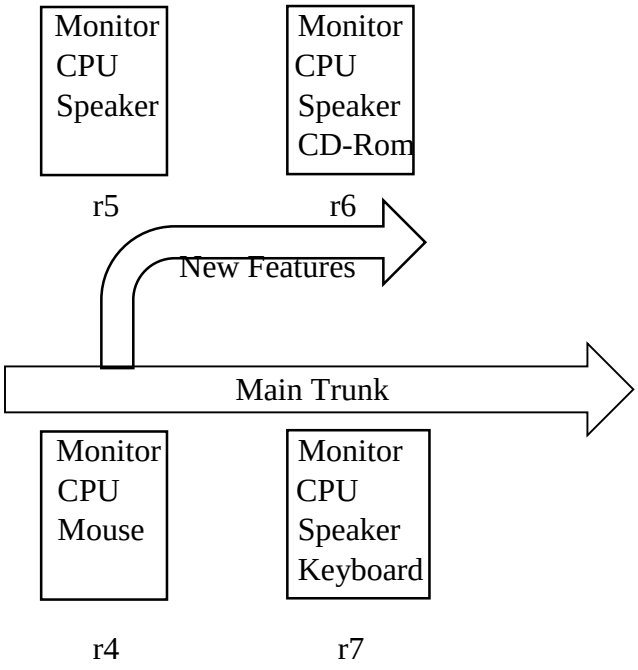
If dislike the changes and want to start over, revert to the previous version is consider and it can start again or stop. When checking out, can get the latest revision by default. If necessary, can specify a particular revision and run as

Svn co list.txt (get latest version)
…..edit file…..
svn revert list.txt (thrown away changes)
svn co –r2 list.txt (check out particular version)

Advanced actions	Explanations	Source code
Branch	Create a separate copy of a file/folder for private use (bug fixing, testing).	git branch
Merge	Apply the changes from one file to another, to bring it up-to-date. For example, merge features from one branch into another.	git merge <branchname>
Conflict	When pending changes to a file contradict each other.	
Resolve	Fixing the changes that contradict each other and checking in the correct version.	
Check out for edit	Checking put an “editable” version of a file. Some version control software have editable files by default, others require an explicit command.	

4. Branching

Branches let user copy code into a separate folder se user can separate it:

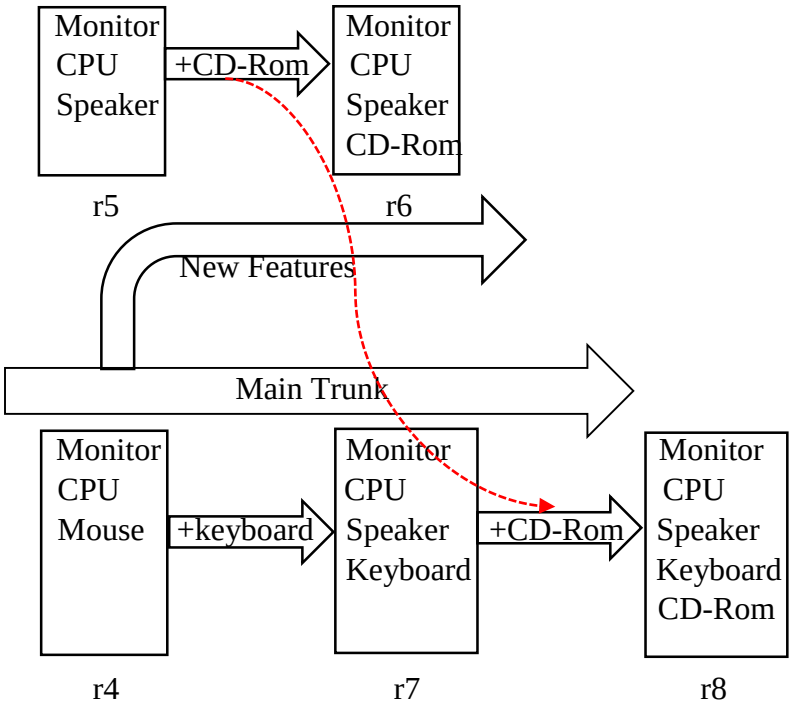


For example, user can create a branch for new, experimental ideas for the above list. Depending on the version control software, creating a branch may change the revision number. User can change the code and work out the links. Since it is in a separate branch, user can make changes and test in isolation. Also, user can create a branch simply by copying a directory to another.

svn copy http://path/to/trunk http://path/to/branch

5. Merging

If we diffed r6 and r7, keyboard feature would lose in main. Imagine get rid of the changes from the experimental branch (+CD-Rom) and adding that to main. Main may have had other changes and it just need to insert the CD-Rom feature.



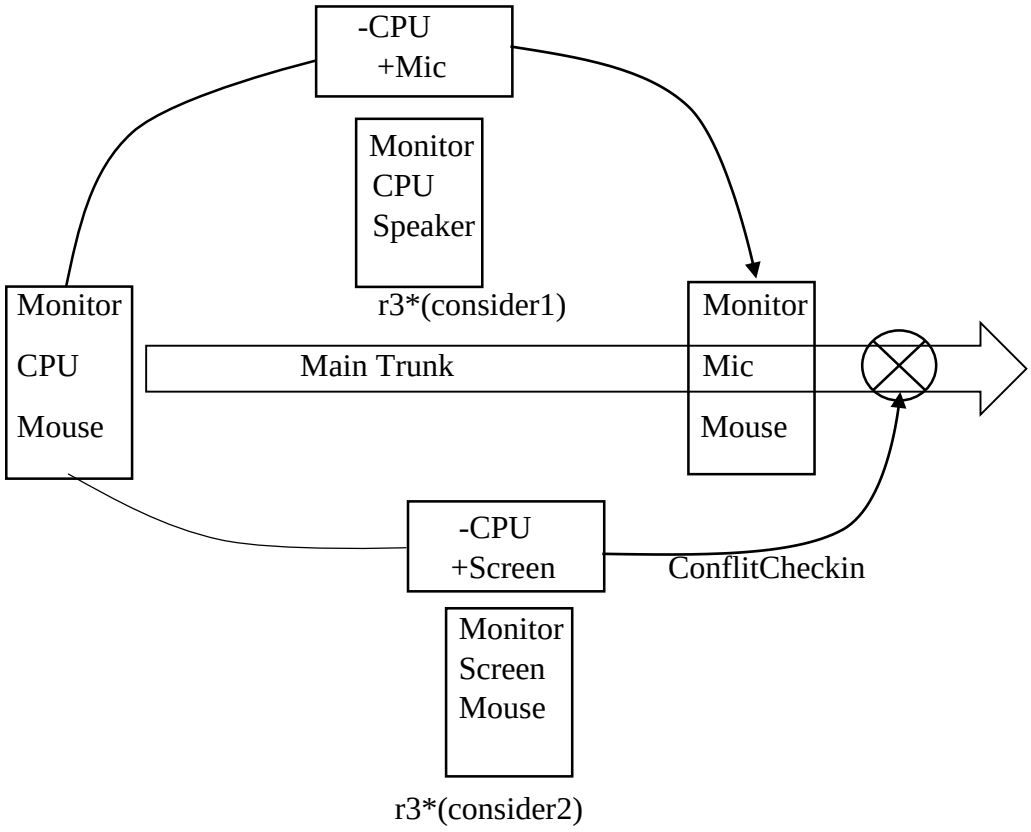
Inside the main trunk, run as:

svn merge –r5: 6 http://path/to/branch

This command diffs r5-r6 in the experimental branch and applies it to the current location. Unfortunately, does not have an easy way to keep track of what merges have been applied so if not careful may apply the same changes twice. It's a planned feature but the current advice is to keep a changelog message means already merged r5-r6 into main.

6. Conflicts

Sometimes, version control software can automatically merge changes to different parts of a file. Conflicts can arise when changes appear. For example: consider want to remove the CPU and replace it with microphone (-CPU, +microphone) and consider want to replace CPU with a screen (-CPU, +screen).



When changes overlap and contradict like this, the version control software may report a conflict and not let to check in — it's up to user check in a newer version that resolves this dilemma. A few approaches:

- Re-apply the changes. Sync to the the latest version (r4) and re-apply the changes to this file: Add screen to the list that already has mic.
- Override their changes with. Check out the latest version (r4), copy over the version, and check the version in. In effect, this removes cheese and replaces it with screen.

7. Why merging is necessary

Merging is necessary in the version control software because creating merge in version control software keeps multiples streams of work independent from each other while also providing the facility to merge that work back together, enabling developers to verify that the changes on each branch do not conflict. Many software teams apply a practice of merging for each feature or perhaps branching for each release or both. There are many different workflows that teams can choose from when the team decide how to make use of merging facilities in version control system.

8. Merge Conflict

In Git, "merging" is the act of integrating another branch into the current working branch which taking changes from another context and combine them with the current working files. A great thing about having Git as the version control software is that it makes merging extremely easy: in most cases, Git will figure out how to integrate new changes.

However, there's a handful of situations where might have to step in and tell Git what to do. Mostly, this is when changing the same file. Even in this case, git will most likely able to figure it out on its own. But if two people changed the same lines in that same file, or if one person decided to delete it while the other person decided to modify it, Git simply cannot know what is correct. Git will then mark the file as having a conflict - which you'll have to solve before you can continue your work.

9. Resolution

Many time, when do git push/pull or git merge, should be end up with conflicts. In most cases, solution to merge conflict is as simple as discard with the local changes or remote/other branch changes. To resolve merge conflicts.

- Find files with merge conflict
  1. Change working directory to project folder.
     cd project-folder.
  2. Search for all conflicting files.

     grep –lr '<<<<<<<'

- Resolve easy/obvious conflicts
  1. At this point may review each files. If solution is to accept local/our version, run:

     git checkout –ours path/file

  2. If solution is to accept remote/other-branch version, run:

     git checkout –theirs path/file

  3. If have multiple files and you want to accept local/our version, run:

     grep –lr'<<<<<<<' .|xargs git checkout --ours

  4. If have multiple files and you want to accept remote/other-branch version, run:

     grep –lr'<<<<<<<' .|xargs git checkout –theirs

10. Repositories

In version control software, there are two different kinds of repositories which are distributed and centralised. Repository is a record of the entire history of the project. When change something in the working copy, then should commit it to the repository. However, distributed repository is selected because distributed version control system is a program that shows the entire history of your project files into a special hidden folder inside the working copy on your hard drive. Every commit could added to the history in that hidden folder. Moreover, it allow to transfer project history between different copies of the repository; each copy contains the entire history and the software knows how to keep the copies in sync even after they have been modified independently. It is common to keep a "master copy" on a server elsewhere which user and the collaborators can use to keep up with each other's work or use to make the work publicly or simply use as a private backup. But no need to have a separate server in order to keep track of the changes. Only the working folder and the version control software which manages that hidden folder of history inside it.

Popular distributed version control systems are Git and Mercurial.