# Version Control System

* Why? 1. Prevent from overwrite 2.undo ..
* Lock-Modify-Unlock? (1. Time consuming 2. DeadLock 3. Forget to Unlock)
* **Copy-Modify-Merge ?** Yes!

Steps:

* + You Check out(copy)/create a version of the project
  + You Modify the project locally, commit to remote repository
  + Your co-worker can merge your changes into their working copies

# Concurrent Version System (CVS)

* main line of the code: truck/main/master
* parallel line of code: branch (to add a feature or try to fix a bug)
* **shortcomings:**
  + no support for moving or renaming ( treat old file as deleted)
  + Branch operations are expensive (Ubuntu os)
  + Commits are not atomic
    - Atomic: means either commit succeeds or go back to the state before commit was applied.
    - Not Atomic: repository might be in a corrupted state

# SubVersion (SVN)

* intended to address the shortcomings of CVS.
  + Atomic commits （commit successfully or not at all)
  + Branch operatings are much less expensive
  + Support for Moving or Renaming
* shortcomings of SVN:
  + Centralized
    - Users all connect to the central repo. If network access is unavailable, none of the source control operations can take place.
  + Slow Speed
    - Both SVN and CVS take long time to update source code

# Centralized repo or Distributed repo?

* Central? (Bad Idea)
  + only one central repository (whick keep project’s history), if you lose the network, you lose everything
* Distributed? (yes! Like Git!)
  + Every developer can view the entire history of the repo offline.
  + Every developer can switch between branches and make commits
  + Operations are dramatically faster (operations are done on the local disk instead of the network)
  + How to use Git? How git works? Refer to a separate doc: “Git”