Problems Working Alone

- Have you ever done one of the following?
  - Had code that worked, made a bunch of changes and saved it, which broke the code, and now you just want the working version back…
  - Accidentally deleted a critical file, hundreds of lines of code gone…
  - Somehow messed up the structure/contents of your code base, and want to just “undo” the crazy action you just did
  - Hard drive crash! Everything’s gone, the day before deadline
- Possible options:
  - Save as (MyClass-v1.java)
    - Ugh. Just ugh. And now a single line change results in duplicating the entire file…

- Who’s computer stores the “official” copy of the project?
  - Can we store the project files in a neutral “official” location?

- Will we be able to read/write each other’s changes?
  - Do we have the right file permissions?
  - Lets just email changed files back and forth! Yay!

- What happens if we both try to edit the same file?
    - Bill just overwrote a file I worked on for 6 hours!

- What happens if we make a mistake and corrupt an important file?
    - Is there a way to keep backups of our project files?

- How do I know what code each teammate is working on?

<u>Solution: Version Control</u>

- Version Control System: Software that tracks and manages changes to a set of files and resources.
- You use version control all the time
    - Built into word processors/spreadsheets/presentation software
        - The magical “undo” button takes you back to “the version before my last action”
    - Wiki’s
        - Wiki’s are all about version control, managing updates, and allowing rollbacks to previous versions
- Many version control systems are designed and used especially for software engineering projects

    - Examples: CVS, Subversion(SVN), Git, Monotone, BitKeeper, Perforce
- Helps teams to work together on code projects
    - A shared copy of all code files that all users can access
    - Keeps current versions of all files, and backups of past changes
    - Manages conflicts when multiple users modify the same file
    - Not particular to source code; can be used for papers, photos, etc.
        - But often works best with plain text/code files

Repositories

- Repository (aka “repo”): A location storing a copy of all files
    - You don’t edit files directly in the repo
    - You edit a local working copy or “working tree”
    - Then you commit your edited files into the repo
- There may be only one repository that all users share (CVS, Subversion)
- Or each user could also have their own copy of the repository (Git, Mercurial)
- Files in your working directory must be added to the repo in order to be tracked

What to put in a Repo?

- Everything needed to create your project:
  - Source Code
  - Build Files
  - Other resources needed to build your project (icons, text, configs, etc.)
- Things generally NOT to put in a repo (these can be easily recreated and just take up space):
  - Object files (.o)
  - Executables (.exe)
  - IDE Configurations (If your company supports more than one IDE)

Repository Location

- Can create the repository anywhere
  - Can be on the same computer that you're going to work on, which might be ok for a personal project where you just want rollback protection
- But, usually you want the repository to be robust:
  - On a computer that's up and running 24/7
    - Everyone always has access to the project
  - On a computer that has a redundant file system (i.e. RAID)
    - No more worries about that hard disk crash wiping away your project

$ git help <verb>
$ git <verb> --help
$ man git-<verb>