Version Control

Class 10

VCSs

- version control system
- revision control system
- a subset of configuration management

Version Control

The management of changes to documents (files) throughout the development process

- every development environment provides VCS support
- every development methodology incorporates VCS

Features

- a history of the evolution of each file
  - ability to re-create any previous version
  - ability to calculate the difference between two versions
  - ability to merge features of different versions
- track time, developer, data, and metadata
- allow distributed and team development
- allow non-synchronous collaboration
- allow symbolic names for distinguished versions or features
- allow parallel concurrent development using branching

Pessimistic Model

- pessimistic (locking) model
  - assumes conflicts will be frequent and hard to resolve
  - enforces conflict prevention
  - a lock is asserted on resources
  - lock gives exclusive access (at least write) to one developer
  - most common in proprietary, centralized IDEs
    - Rational
    - Microsoft

Optimistic Model

- optimistic (merging) model
  - assumes conflicts are rare and easy to resolve
  - allows simultaneous modifications
  - automatically merges non-conflicting changes
  - detects and flags conflicts
  - provides tools to help resolve conflicts
  - most common in distributed development methodologies
  - universal in open-source development
    - git
    - subversion

Systems

- SCCS 1972 on IBM mainframe systems, ported to AT&T's System III Unix
- RCS 1982 in Berkeley Unix to avoid AT&T's copyrights
- CVS 1986 incredibly simple, robust, powerful superset of RCS
- Subversion (SVN) 2004 from Apache corrected and enhanced some of the most annoying things about CVS; still in very wide use

- all required a central server for multi-person development

git

- Linus Torvalds 2005

git

- Linus Torvalds 2005
- lost free use of proprietary BitKeeper system being used for Linux kernel

git

- Linus Torvalds 2005
- lost free use of proprietary BitKeeper system being used for Linux kernel
- three design principles

git

- Linus Torvalds 2005
- lost free use of proprietary BitKeeper system being used for Linux kernel
- three design principles
  - use CVS as the example of what not to do; if in doubt, make the exact opposite design decision

git

- Linus Torvalds 2005
- lost free use of proprietary BitKeeper system being used for Linux kernel
- three design principles
  - use CVS as the example of what not to do; if in doubt, make the exact opposite design decision
  - support offline distributed workflow with no central server

git

- Linus Torvalds 2005
- lost free use of proprietary BitKeeper system being used for Linux kernel
- three design principles
  - use CVS as the example of what not to do; if in doubt, make the exact opposite design decision
  - support offline distributed workflow with no central server
  - provide strong safeguards against corruption, accidental or malicious (a critical property)

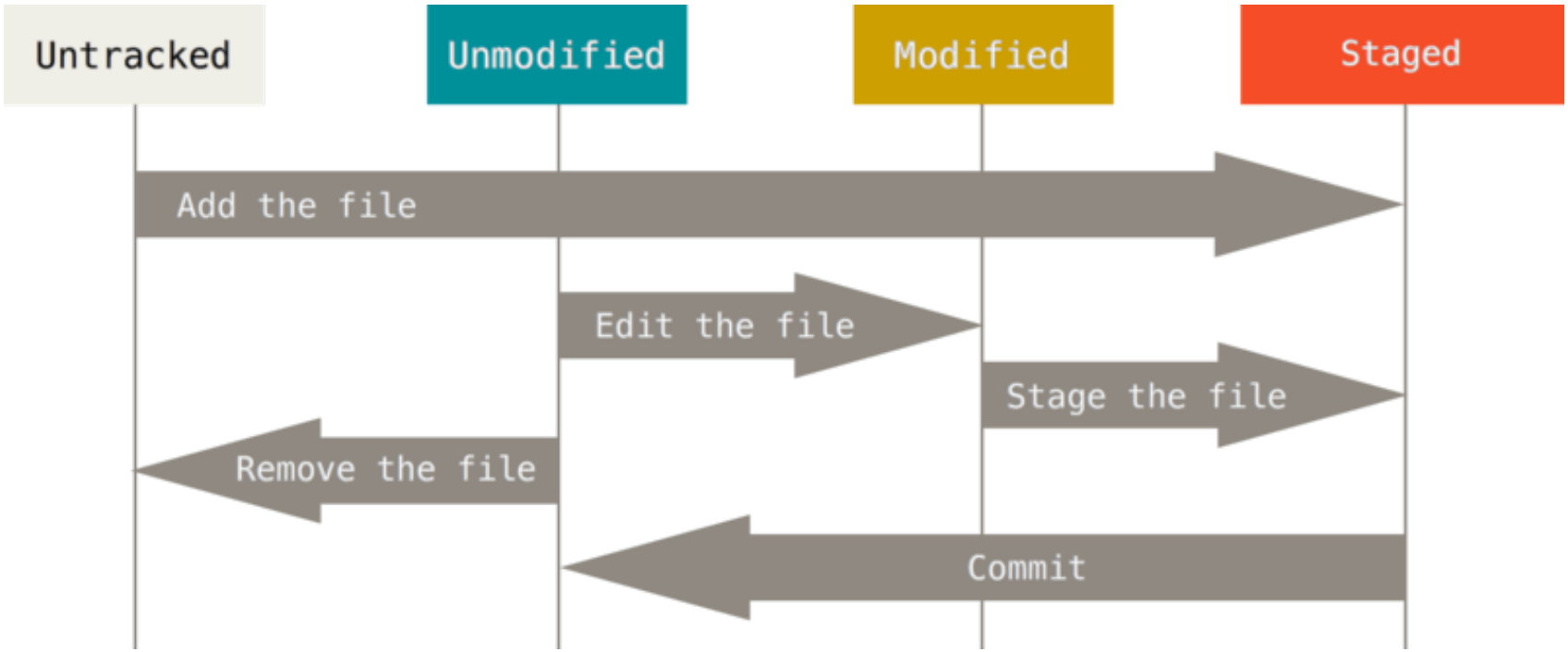
git

- Linus Torvalds 2005
- lost free use of proprietary BitKeeper system being used for Linux kernel
- three design principles
  - use CVS as the example of what not to do; if in doubt, make the exact opposite design decision
  - support offline distributed workflow with no central server
  - provide strong safeguards against corruption, accidental or malicious (a critical property)
- first code 3 April; used for 2.6.12 kernel 16 June
- took a long time to gain popularity, but now overwhelmingly used in agile development

Workflow

- git can be used peer-to-peer or client-server
- we will use git in the client-server mode
- does not require continuous network connection

Local Document States



Resource

- `http://git-scm.com/book/en/v2/`
- read 1.1 – 1.6 and 4.3 by next Thursday
- in this course, we will not be using GitHub, so don't worry about making a GitHub account (although you probably want one for other purposes)
- we will be using ice as our git server

Install

- decide whether your git client will be your own computer or will be ice (you can choose others later)
- if you install on your own computer, follow the instructions in git-scm section 1.5 for your platform
- if you're on windows, I strongly recommend the GitHub version
- if you're on Mac, git may already be installed; check by opening Terminal.app and entering `$ git --version`
- if command not found, or if the version is pre 2.x, you need to install a newer version as discussed in git-scm
- if you're on Linux, any version 2.x or newer is fine (sudo apt install git)

Setup

- now do all the first-time setup steps in git-scm section 1.6

SSH Keys

- you need to generate an ssh RSA public-private key pair on your git client computer (if you don't already have one)
- follow the instructions in git-scm section 4.3 and generate an ssh keypair without a passphrase (just press enter when prompted for a passphrase)
- this will generate two files named id-rsa.pub (the public key) and id-rsa (the private key) in a directory named .ssh
- you need to submit the public key file to the 370 homework submission page by start of class 19 February

Schedule

- see the course calendar for other upcoming stuff