Version Control

Lab Schedule

- Today
  - **Lab 2 – Version Control**
- Next Tuesday (Sept 10th)
  - **Lab Report for Lab 1 due by 11:59pm**
    - Submit via Sakai
  - **Intro to C (for C++ programmers)**
  - **Build Tools + Makefiles**
- Next Thursday (Sept 12th)
  - **Lab 3 – C Programming**
  - **Lab Report for Lab 2 due by 11:59pm**

Pre-Lab

- **Any problems encountered creating repository?**

Before Version Control

1. <Report.doc>
2. <Report.doc.bak>
3. <Report-1.doc>
4. Email off to partner…
5. <Report-2.doc>
6. Partner responds with doc (that is missing the changes you just made)
7. <Report-2a.doc>
8. <Report-2a-WITH-REFERENCES.doc>
9. Email off to partner… Partner responds with new doc <Report-3.doc>
10. <Report-3-FINAL.doc>
11. <Report-3-FINAL-OOPS-FIXED-TYPO-FINAL.doc>

Version Control Features

- Project history tracking
- Concurrent file editing (merges)
- Non-linear program history (branches)
- Naming scheme for program releases (tags)

Motivation for Version Control

- **Why would a single programmer (working alone) use version control?**
  - Backup files
  - Roll-back to earlier (working) version
  - See changes made between current (broken) code and earlier (working) code
  - Maintain multiple versions of a single product
  - Experiment with a new feature
    - Try a risky change in a "sandbox"
    - If it works, you can merge it into the regular code. If it fails, you can throw it away.

Motivation for Version Control

- **Why would a small group of developers use version control?**
  - All the reasons a single programmer would, plus…
  - Merging different changes made by different developers into the same file
    - Add a new function at the bottom? Safe to automatically merge in
    - Re-write a function at the same time another developer is also editing it? Version control will catch this and ask you to decide which edits should "win"
  - Blame – who wrote this buggy code?!?

Motivation for Version Control

- **Why would a large group of developers use version control?**
- Different question: What would it have been like to develop the Linux kernel, Adobe Photoshop, Google Chrome, etc… using:
  - A single shared "folder of code"?
  - Emailing code snippets between developers?
  - Everyone sitting around and sharing one keyboard?

Version Control Basics

- **What kind of files should I keep in version control?**
  - Program source code *(obviously)*
  - VHDL / Verilog files (from digital design class)
  - Matlab scripts
  - HTML files
  - Server configuration files
    - Imagine you work at Livermore National Labs, and your job is to manage Linux cluster computers with 100,000+ machines (nodes)…
  - **Anything that is plain text!**

Version Control Basics

- **What kind of files should I <u>not</u> keep in version control?**
  - *These aren't "rules", so much as "guidelines"…*
  - **Binary data**
    - How do you *merge* two different binary files together? No general-purpose way to do this
  - **Anything auto-generated by the compiler**
    - Object files or executable file
    - Wastes space on useless junk that can be re-created automatically
  - **Text editor temp files** (e.g. `main.c~`)

Version Control Basics

- **Big risk in putting the executable in version control**
  - If you forget to compile before a commit, the executable may not be **in sync** with the attached source code!
  - **Big headache if you ever roll back to this version!**
- **In ECPE 170, all our executable files can be produced in under 5 seconds with one command. There's no need to include them in your repository**

Distributed Version Control

- **Why do they call Mercurial a distributed version control system?**
  - Conventional systems (e.g., Subversion or "svn") have a centralized server hold the "master" copy
  - Distributed version control – each copy is its own full-fledged master! (But you can still push changes from one person's copy to another)
    - Allows version control to work offline
    - Allows version control to work with ad-hoc groups

Version Control in ECPE 170

- Version control **required** for this class
  - Used to distribute boilerplate code for labs
  - Used to turn in assignments when finished

Version Control in ECPE 170

- *If you only do one check-in at the very end of your project, you've missed the whole point of version control, and turned a valuable tool into an obstacle to completing the assignment*

- **Check-in code on a regular basis!**