15-440 Recitation 1
SVN and Makefiles

Alex Katkova
Jason Zaman

CMU CS Spring 2010

Announcements

- Your first project is due February 10th!
- That is a Wednesday, so you have time to visit office hours before the due date and after in case you need to use a late day.
- Start early!
- Ask questions early!

Recitation Mechanics

- These are your recitations
  - We've got a schedule. It's flexible.
  - Ask questions, make comments...
  - 1 part lecture, 1 part “public office hours” (homework or project questions? Go for it!)

Recitation Overview

- Today: Intro and Revision Control
- Makefiles
- Debugging
- Some project info

Revision Control

- Before you write a line of code...
- Use subversion/CVS/git/etc
- Provides access to all old versions of your code
  - No more “cp file.cpp file.cpp.2010-01-29-oh-god-please-let-this-work”

What is revision control?

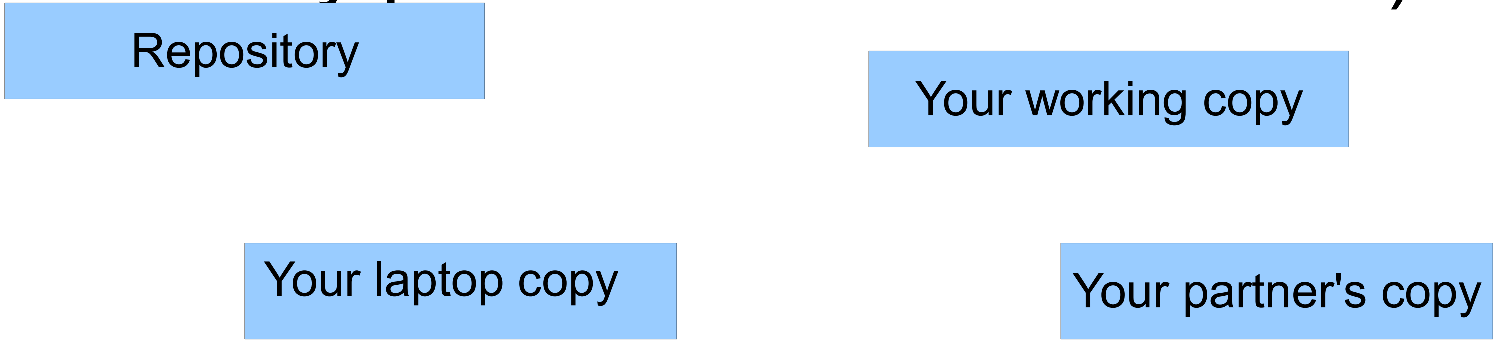
- A repository that stores each version
- You explicity “check out” and “check in” code and changes

Why do I want it?

- Super-undo: go to arbitrary versions
  - you've managed to delete all your code? No problem.
- Track changes
- Concurrent development
- Snapshots
  - Turning in the assignment: just make a snapshot of your code and we will grade that snapshot. You can keep developing afterwards.

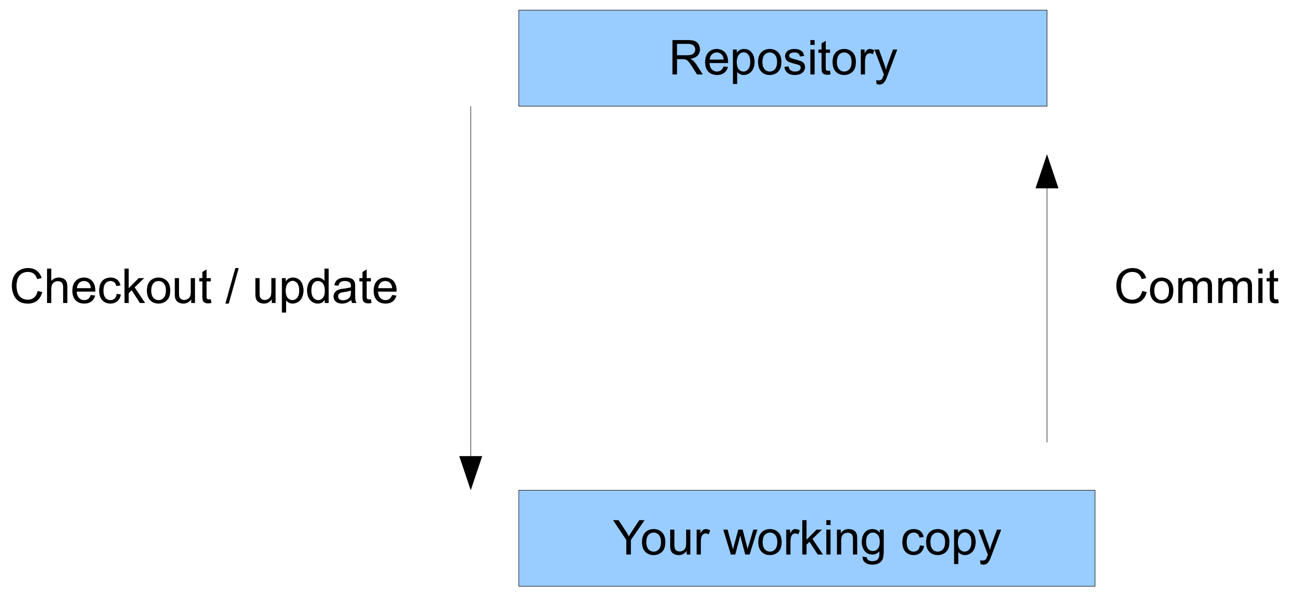
The repository

- Master copy of the code is separate from what you work on
- You can have multiple working copies checked out (so can any partners or team members)

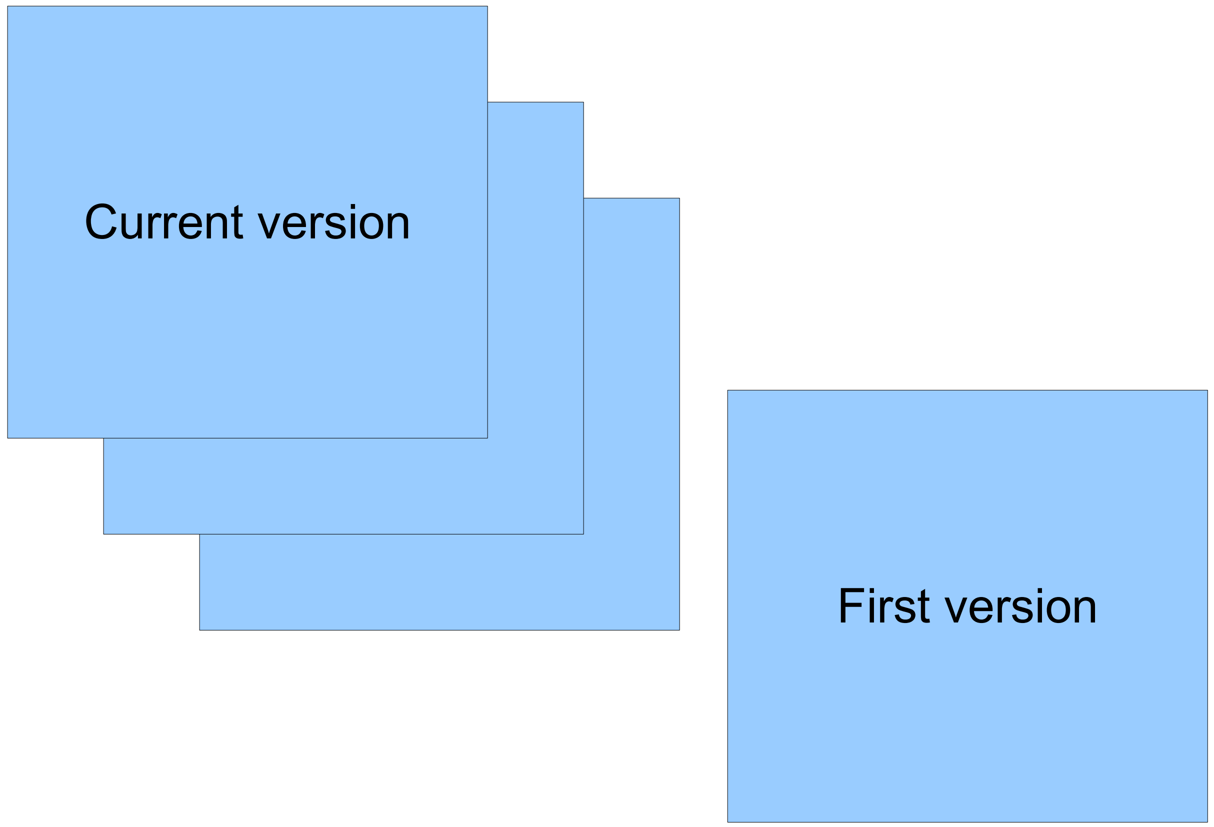


Check out and commit

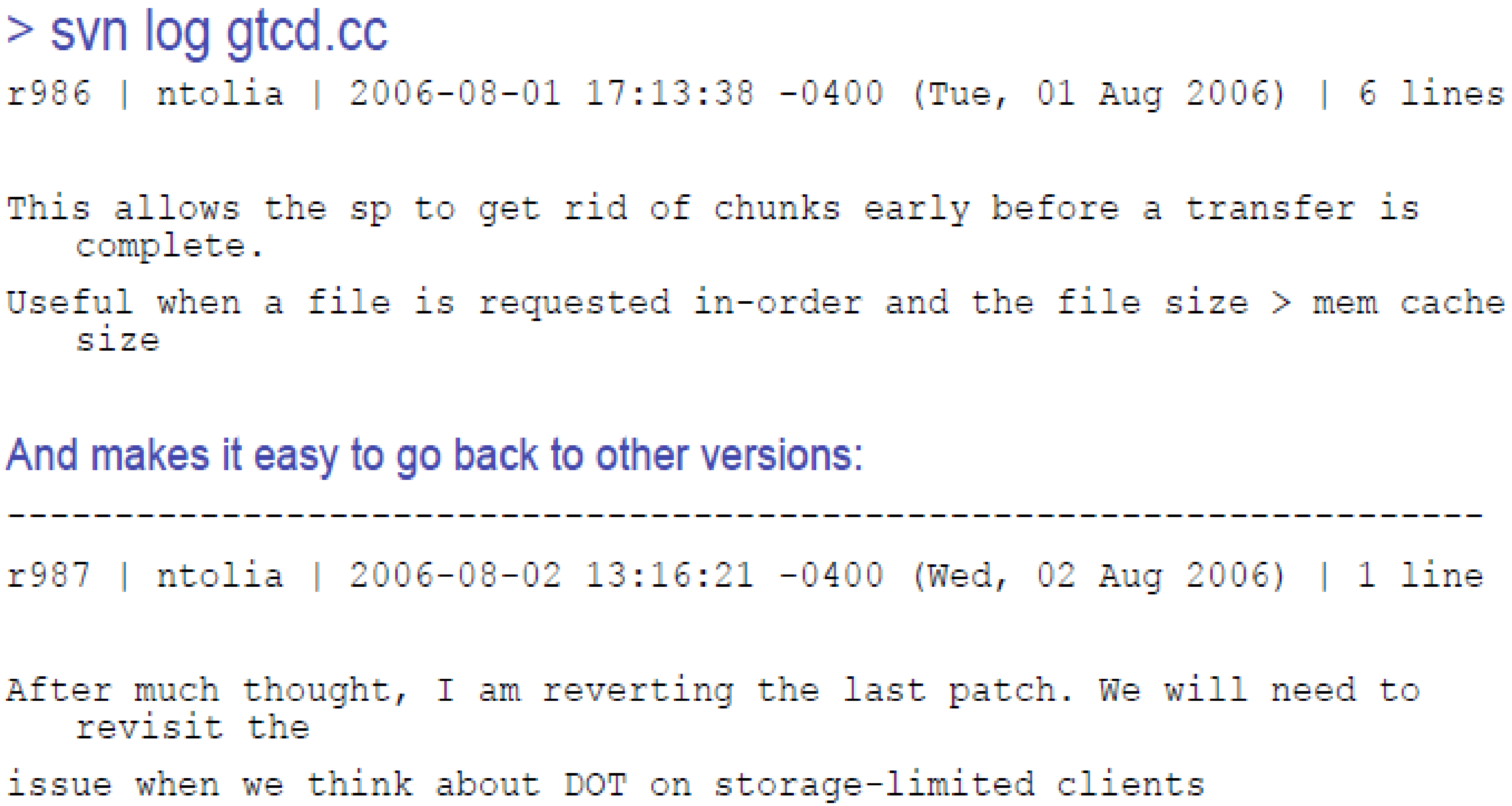
- Explicitly synchronize with the repository



Every revision is available



And you can see what changed



> svn log gtcd.cc

```
r986 | ntolia | 2006-08-01 17:13:38 -0400 (Tue, 01 Aug 2006) | 6 lines

This allows the sp to get rid of chunks early before a transfer is
   complete.
Useful when a file is requested in-order and the file size > mem cache
   size
```

And makes it easy to go back to other versions:

```
------------------------------------------------------------------------
r987 | ntolia | 2006-08-02 13:16:21 -0400 (Wed, 02 Aug 2006) | 1 line

After much thought, I am reverting the last patch. We will need to
   revisit the
issue when we think about DOT on storage-limited clients
```

Concurrent Development

- Each person checks out a copy
- Both can work at the same time without much fear of clobbering the other with a heavy club
  - changes are only visible on commits and updates
- What happens if both people edit a file at the same time and commit?

Possibilities

• If Alice and Bob edit different parts of the file, their versions will likely be successfully merged. Yay SVN magic!

• If Alice and Bob's changes overlap, they will get a conflict.

Resolving Conflicts

- Subversion will give you 3 files:
  - the original with conflict markers (<<<<)
  - the version you were editing
  - the latest version in the repository
- You can do several things:
  - keep your changes, discarding others
  - toss your changes
  - manually resolve

Branches

- Multiple paths of development

  - Release 1.0 only gets security patches

  - “development” branch gets everything

- “tags” or “snapshots”

  - save a good known state

- Merging branches : read on your own

Subversion commands

- svn checkout https://moo.cmcl.cs.cmu.edu/440/..
- svn commit
- svn update (svn up)

- svn add
- svn mkdir

- svn copy (create a branch or snapshot)
- svn diff (see the difference between two versions)

Sample walkthrough

```
> svn checkout https://moo.cmcl.cs.cmu.edu/440/Project...
    A    trunk/
    …
> cd trunk
> echo "#empty Makefile" >> Makefile
> svn add Makefile
  A         Makefile
> svn commit
[svn will open an editor for log message]
Adding          Makefile
Transmitting file data ..
Committed revision 2.
```

Turning stuff in

```
> svn add server.cpp
    A    trunk/server.cpp
    …
# tested, it works!

> svn copy trunk tags/final
  A          tags/final
> cd tags; svn status
  A  +   final
# test your code in the final directory!
> svn commit
[svn will open an editor for log message]
...
Transmitting file data ..
```

Some additional thoughts

- Update, make, test, then commit
- Always update before starting work (just in case)
- Try not to break the checked in copy

  - making a lot of scary changes? Use a branch
- Don't use svn lock
- Revision control will save you lots of pain!!!

Makefiles!!!!

Simple g++

- If we have files:
  - prog.cpp – the main program file
  - lib.cpp – library .cpp file
  - lib.h – library header file

- g++ -c prog.cpp -o prog.o
- g++ -c lib.cpp -o lib.o
- g++ lib.o prog.o -o binary

g++ flags

- -g : for debugging (so that gdb will show you line numbers)
- -Wall : all warning
- -Werror : treat warnings as errors

Don't repeat yourself!

```
% g++ -g -Wall -Werror -c prog.cpp -o prog.o
% g++ -g -Wall -Werror -c lib.cpp -o lib.o
% g++ -g -Wall -Werror lib.o prog.o -o binary
```

```
CXX = g++
CFLAGS = -g -Wall -Werror
OUTPUT = binary
```

In general for a Makefile

```
target: dependency1 dependency2 …
	Unix command (start line with a TAB)
	Unix command

g++ lib.o prog.o -o binary

binary: lib.o prog.o
	g++ lib.o prog.o -o binary
```

Example

```
binary: lib.o prog.o
	g++ -g -Wall lib.o prog.o -o binary

lib.o: lib.cpp
	g++ -g -Wall -c lib.cpp -o lib.o

prog.o: prog.cpp
	g++ -g -Wall -c prog.cpp -o prog.o

clean:
	rm *.o binary
```

Project 1!!!