CSE 303
Concepts and Tools for Software Development

Magdalena Balazinska
Winter 2007
Lecture 19 – C++: Templates and STL
Tools: Version control

Where We Are

- We are almost done talking about C++
  - Still need to talk about templates and STL
- So what are we going to do for the rest of the quarter?
  - More tools: version control (today)
  - Software engineering basics
    - Unit testing, stubs, specifications
    - Writing robust and readable code
  - Societal implications
  - A few extra things: threads and (maybe) profilers

Introduction to Templates

- Motivation: often want to perform the same operations on different data types
- Example: storing data in a linked list
  - Solution 1: Create a new list class for each data type we want to store in a list
  - Solution 2: Force all data types to have a common ancestor X and create a list of X (Java solution)
  - Solution 3: Create a generic list class, and have the compiler use that generic class as a *template* to generate code for all the list classes we need
    - Note: this is DIFFERENT from Java generics

C++ Templates Basic Idea

- With a single code segment, define a whole group of related *functions* or *classes*
- From the template, the compiler **generates** the code for all actual functions or classes
  - C++ templates are said to be implemented “by expansion”
- The generated code is then compiled

Syntax for Class Templates

- Class definition in .h file

```
template < class T >
class MyClass {
  // Here use T like ordinary type
  bool test(T item);
};
```

- Function definitions in the .cc file

```
template < class T >
bool MyClass<T>::test(T item) {
  // here use T like ordinary type
};
```

Syntax for Using Class Templates

```
MyClass<int> example1;
example1.test(3);

MyClass<char> example2;
example2.test('b');

...
```

- Full example in file `template.cc`

Standard Template Library

- C++ library of:
  - Basic data structures (i.e., container classes)
    - Lists, Maps, Sets, etc.
  - Iterators for traversing these containers
    - Iterators are a generalization of pointers
  - And basic algorithms to operate over various containers: sort, reverse, etc.
    - Algorithms are decoupled from specific containers
    - They are templates parameterized by the type of iterator
- We will only consider two concrete examples
  - `list` in lecture and `map` in assignment

Example: List of Integers

```
#include <list>
[...]

  list<int> my_list;
  for ( int i = 0; i < 10; i++) {
      my_list.push_back(i);
  }

  list<int>::const_iterator i;
  for ( i = my_list.begin();
        i != my_list.end(); ++i ) {
      cout << “Element is “ << (*i) << endl;
  }
```

- Other example in file `main.cc`

Java Generics

- Very different from C++ templates and STL
  - Ex: generic collections classes are based on std Java collections classes where everything is a container of Objects
- Java generics are implemented by “type erasure”
  - Compiler reads type information
  - Compiler performs type checks
  - Compiler automatically generates type casts
  - Compiler erases any type information
  - So the resulting bytecode is the same as without using generics, but traditional collections classes
- Goal in Java was backward compatibility

No Templates nor STL on Final

- Templates and STL are an advanced topic
- We overview them briefly because they are very frequently used in C++
- But there will be no question about templates nor STL on the final

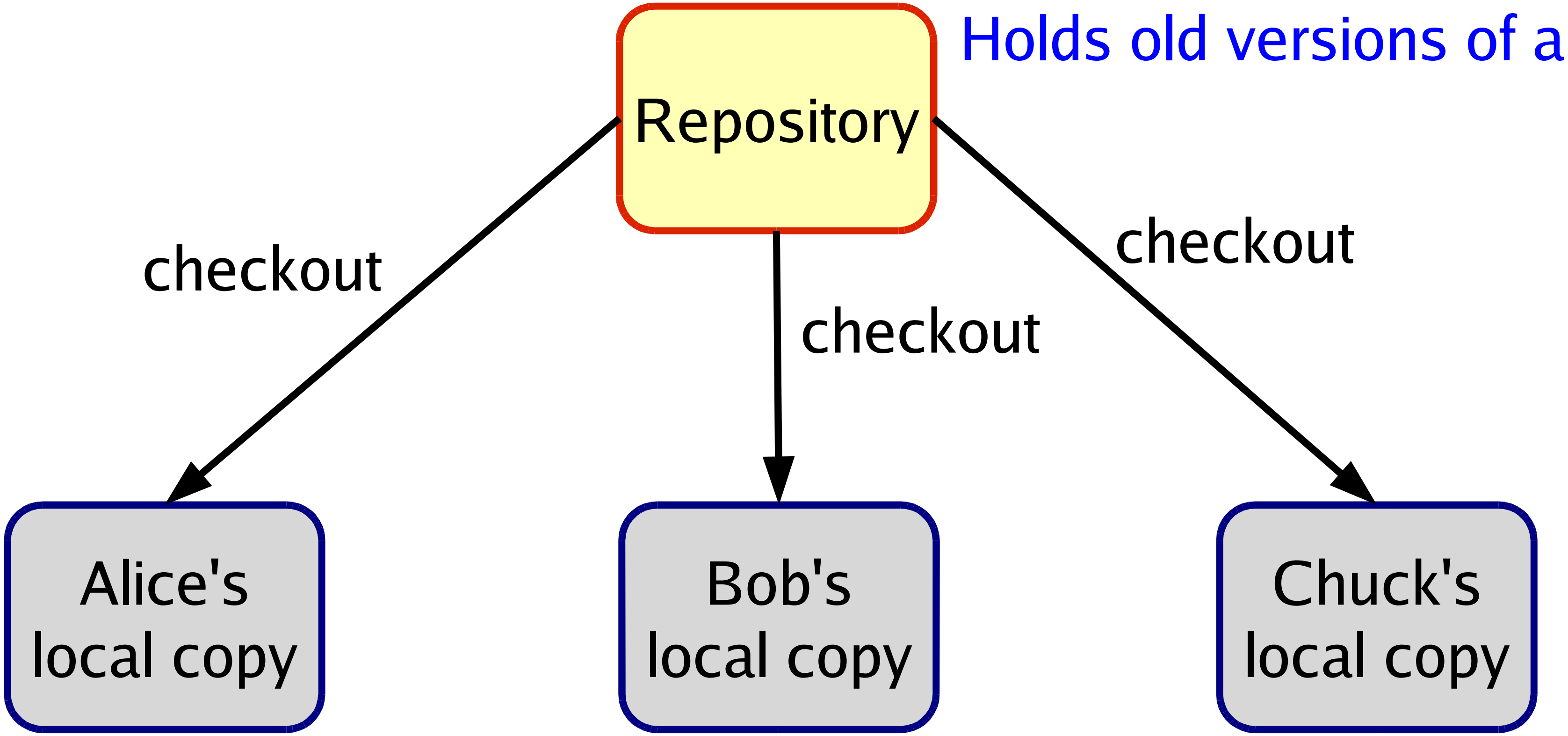
Version Control Systems: Motivation

- Alice, Bob, and Chuck are working on a large software system
  - Where should they keep their source code?
  - What if they want to work on their laptops? from home? disconnected from the network?
  - How should they manage concurrent modifications?
  - What if Bob needs to keep the code stable to give a demo while Chuck would like to try a new idea?
  - What if Chuck tries his new idea and breaks the code the day of the demo?

Version Control System

- Goal of a version control system
  - Handle simultaneous concurrent changes
  - Manage multiple versions of a system
- Many version control systems exist
  - CVS, RCS, Subversion, SourceSafe, ClearCase
- Just like any other tool that we study
  - All these tools have similar goals and similar basic features (but different ways to use these features)
- CVS can manage any files, not just source code
  - I use it for everything... including course materials

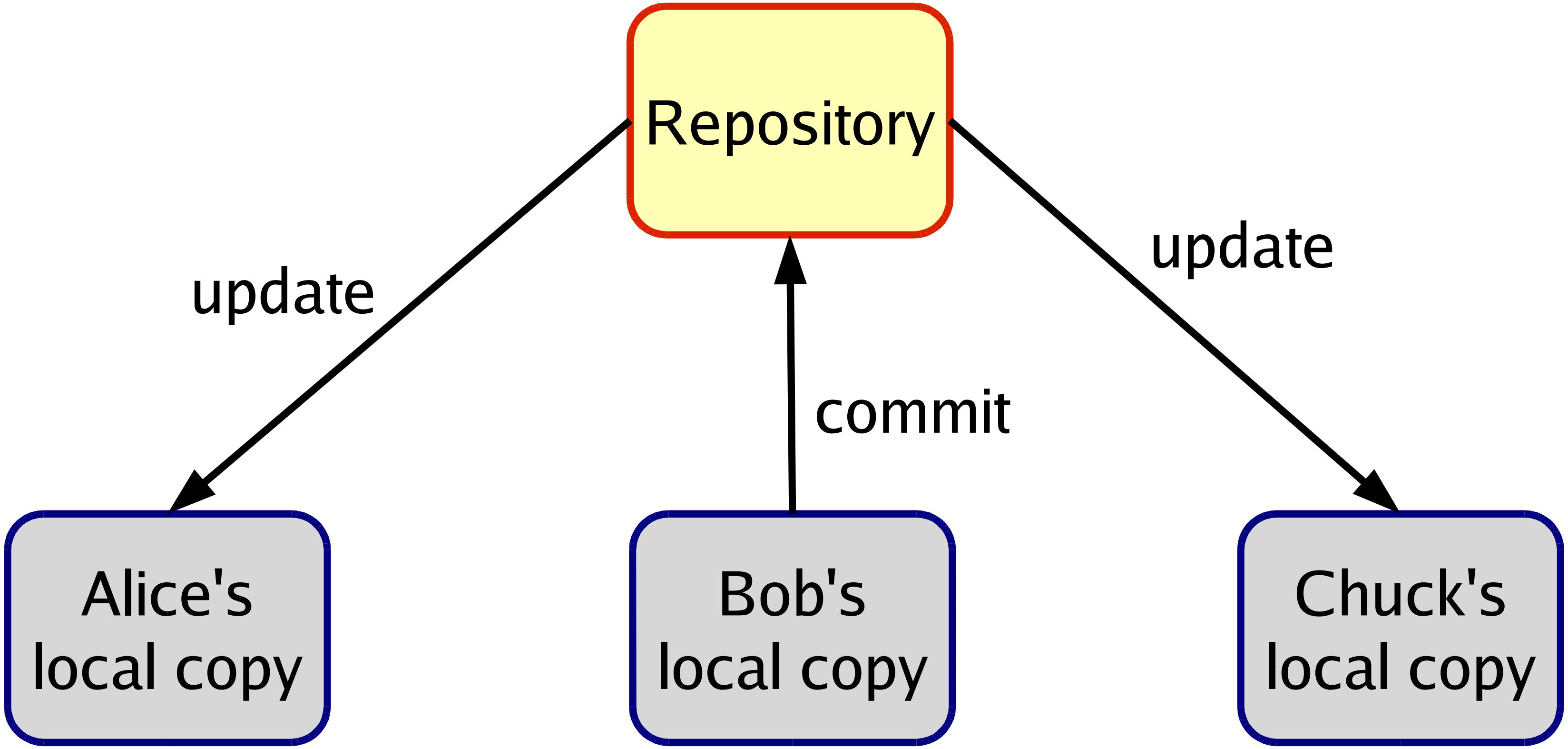
CVS: Basic Idea



Holds master copy of all files
Holds old versions of all files

Developers should NOT modify the repository directly
Instead, each developer checks out and modifies a working copy

CVS: Basic Idea



Modifies files
Adds files
Adds directories

Basic Idea Summary

- There exists one CVS repository
  - Holds the master copy of all files for **all projects**
- Each software developer
  - Checks-out a local copy of the files for a project
  - Modifies the files in the local copy
  - Commits his/her changes periodically
  - Updates his/her local copy periodically
    - To see changes made by other developers
  - Adds new files that he/she creates
- Developers use the CVS program to interact with the repository and perform the operations listed above

What Goes Into CVS

- In general: keep in repository ONLY what you need to build the application
  - Never add files that are generated automatically
  - Yes: .cc, .c, .h, Makefile
  - No: .o files or executable
- Think before you add a file to CVS
  - Although you can always remove it later if you make a mistake or if you change your mind

Basic CVS Commands

- Set-up a repository (this is done only once)

```
cvs -d /dir/of/cvsroot init
```

- Add a new project to the repository (once per project)

```
cvs -d /dir/of/cvsroot import pname owner tag
```

- Working on a local copy (frequent commands)

Create local copy: `cvs -d /dir/of/cvsroot co pname`

Commit changes: `cvs com .`

Update local copy: `cvs up -d .`

Add a new file or directory: `cvs add file`

Add a binary file (ex image): `cvs add -kb file`

Log Messages

- Commit messages are mandatory
  - -m “short message”
  - -F filename-with-long-message
  - Else an editor pops up
    - Write your message
    - Save and quit
- Default editor: vi
  - Press “`i`”, write message
  - Press “`ESC :wq ENTER`”
- You can change the default editor

Possible to setup CVS to send out email (with the log message) after each commit

Other Useful CVS Commands

- Described in CVS documentation
  - `http://ximbiot.com/cvs/wiki/`
- Some frequently used commands
  - View commit history of a file
  - View differences between revisions
  - Get version of files as of some date in the past
  - Remove a file
  - Tag a version of all files
  - Create a new branch
  - Merge changes between branches

Working with CVS

- Generic structure of a CVS command

```
cvs cvs-options cmd cmd-options filenames/dirnames
```

- Environment variables (there are more)
  - CVSEDITOR: editor to use for log messages
  - CVSROOT: location of cvs repository
    - I often don't use it and specify -d option when first checking out a project
  - CVS_RSH: must be set to ssh when trying to access repository remotely

```
cvs -d login@server:/dir/of/cvsroot cmd ...
```

Conflicts

- When many people edit the same files at the same time, conflicts can occur
- CVS tries to merge changes automatically
  - Uses `diff` and `patch`
  - Merging is **line-based**
    - (`-kb` prevents `cvs` from trying to merge changes)
  - Conflicts indicated in working copy
    - Search for <<<<<
  - When in doubt
    - Make a copy of your local files before updating!
- Some tools enforce locking but CVS does not

There Is Little Magic to CVS

- The repository just uses directories and files
  - Repository must have correct group permissions
- Files are kept in terms of diffs
  - So small changes lead to small increase in repository size
- Files are kept read-only to avoid “mistakes”
  - cvs commands temporarily change permissions
- cvs commands also temporarily lock repository
  - Locks can stick around if cvs commands are interrupted, so be careful
  - But you can remove left-over locks manually

Summary

- Version control system such as CVS
  - One of the key software development tools
  - All companies use them!
- Advantages
  - Much better than manually emailing files, adding dates or version numbers to files, etc.
  - Handles concurrent changes
  - Manages multiple versions
  - Remembers old versions
  - Useful for software but works on any files!

Readings

- Carefully study the code that accompanies today's lecture

- Standard Template Library Reference
  - `http://www.sgi.com/tech/stl/`

- Online CVS documentation
  - `http://ximbiot.com/cvs/wiki/`
  - manpage for cvs is also helpful