Version Control Basics with Subversion

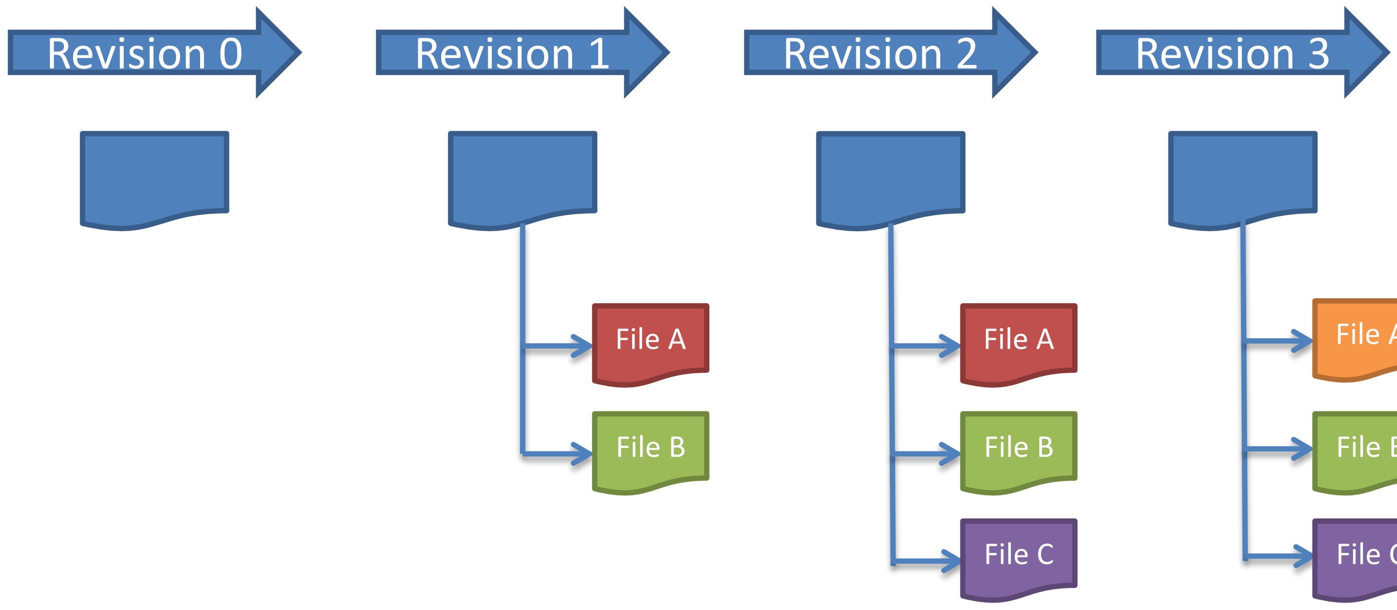
Le Yan

User Services

HPC @ LSU

CENTER FOR COMPUTATION & TECHNOLOGY

What is Version Control



What is Version Control

- The purpose of version control is to manage the changes made to the files that belong to a code base
  - Also know as source control or revision control
- Starting from the initial set of files, each subsequent revision or change is marked with a number or letter code, along with the time stamp and the name of the contributor

Benefits of Version Control

- Make collective code development easier
- Continuous, incremental backup of the code
- Can roll back to older versions of a code base
- Can track bugs, timeline and releases

What is Repository

- Repository: the place where all changesets are stored
- Common layout:
  - Trunk: the main code version
  - Branches: copied of code that can be modified
  - Tags: snapshots of code that are never changed

What is Subversion

- A non-distributed version control system
  - One canonical repository
  - All changes are submitted to the repository
  - All changes are retrieved from the repository
- There are a few others
  - Git
  - Hg Mercurial

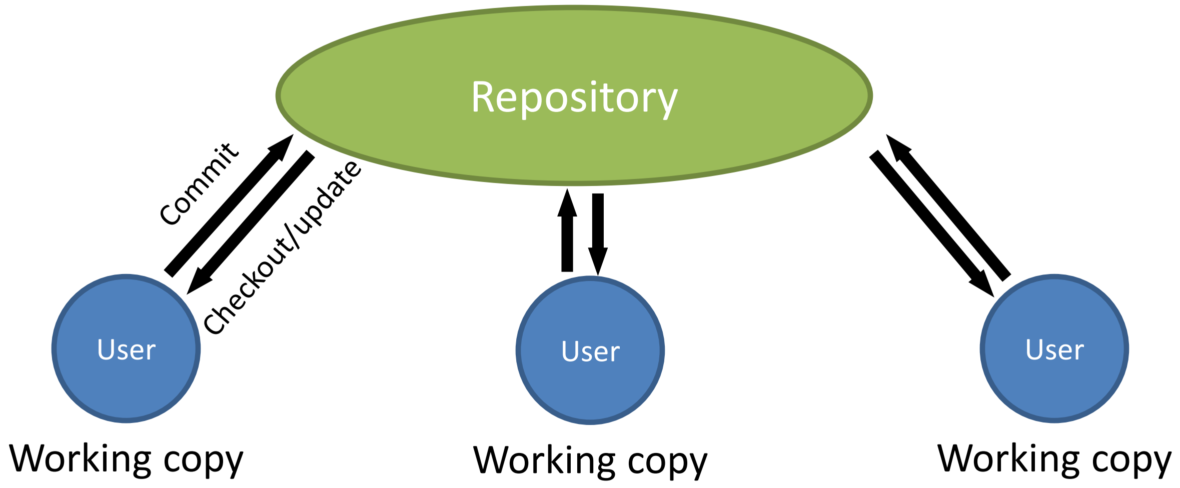
How Subversion Works

Server

- The repository is created and maintained by the administrator

Client

- Users check out their local working copy (happens only once) and make and commit changes



LSU CENTER FOR COMPUTATION & TECHNOLOGY

Installing Subversion

- Linux and Mac users
  - `svn`: the command line client tool
    - `svn <command> <options>`
  - `svnadmin`: admin’s tool to create and maintain a repository
  - Chances are they are already there
- Windows users
  - TortoiseSVN
  - Subclipse
  - RapidSVN

CENTER FOR COMPUTATION & TECHNOLOGY

Setting Up Repository

- Command line: `svnadmin create`
  - Protocols: local file system, svn+ssh, http/s
- Online repository hosting services
  - Google project hosting, Github (using git)…

```
[lyan1@lyan1-1 workspace]$ ls -l
total 0
[lyan1@lyan1-1 workspace]$ svnadmin create repos
[lyan1@lyan1-1 workspace]$ ls repos/
conf   dav   db   format   hooks   locks   README.txt
```

CENTER FOR COMPUTATION & TECHNOLOGY

Import

- `svn import <path to files> <URL>`: import files into the initial repository

```
[lyan1@lyan1-1 workspace]$ svn import /home/lyan1/code/Solutions
file:///home/lyan1/workspace/repos/ -m "initial import"
Adding         /home/lyan1/code/Solutions/precis.f90
Adding         /home/lyan1/code/Solutions/pi.f90
Adding         /home/lyan1/code/Solutions/laplace_solver.f90

Committed revision 1.
```

Basic User Workflow (1)

- Check out a local working copy (only happens once)
  - `svn co` or `svn checkout`
- Update own working copy from the repository
  - `svn up`
- Make changes to the working copy
  - `svn add`, `svn delete`, `svn copy`, `svn move`

CENTER FOR COMPUTATION & TECHNOLOGY

Basic User Workflow (2)

- Examine the changes and undo the change if necessary
  - `svn status`, `svn diff`, `svn revert`
- Resolve conflicts (merge others' changes)
  - `svn resolved`
- Commit the changes
  - `svn commit`
- Display help message
  - `svn help <command>`

Checking Out a Working Copy

- `svn co <URL>`

```
[lyan1@lyan1-1 fortranworkshop]$ ls
[lyan1@lyan1-1 fortranworkshop]$ svn co
file:///home/lyan1/workspace/repos
A    repos/precis.f90
A    repos/pi.f90
A    repos/Pieces
A    repos/Pieces/set_bcs.f90
A    repos/Pieces/params.f90
A    repos/Pieces/main.f90
A    repos/Pieces/modern.f90
A    repos/Pieces/params.mod
A    repos/Pieces/Makefile
A    repos/Pieces/initialize.f90
A    repos/Pieces/laplace.f90
A    repos/laplace_solver.f90
A    repos/save.f90
Checked out revision 1.
```

CENTER FOR COMPUTATION & TECHNOLOGY

Revisions

- Revision numbers are global across the whole repository
- A commit creates a snapshot of the entire tree at that revision number
  - No additional space needed for files that are not affected by the revision

Update Working Copy

- `svn up` - updates the local files to match the repository
  - Need to cd the working directory (the local directory that you have checked out from the repository)
  - `-r` option: go to a particular older revision
  - `-r <version> <file name>`: get an older revision of certain file

Update Working Copy

- Each updated item occupy a line which starts with a character reporting the action taken
  - "A" – Added
  - "C" – Conflicted
  - "D" – Deleted
  - "G" – Merged without a problem
  - "U" - Updated

```
[lyan1@lyan1-1 repos]$ svn up
D    void.f90
A    another.f90
Updated to revision 7.
```

Exercise 1

- Set up your own repository, import some files and check out a working copy (to a different location)
  - `svnadmin create`
  - `svn import`, or
- Check out a working copy from the repository hosted by Google Project Hosting (googlecode.com password: **Nc5gz9bu5CV6**)
  - `svn co https://hpc-workshop.googlecode.com/svn/trunk/ --username lsuhpchelp@gmail.com`

LSU CENTER FOR COMPUTATION & TECHNOLOGY

Making Changes - Add

- `svn add <file name>`
  - Add files, directories and symbolic links to the repository
  - When a directory is added, everything under it will be added as well, unless the –non-recursive (-N) option is used

Making Changes - Delete

- `svn delete <file>`
  - Delete files, directories and symbolic links from the repository
  - Files and links will be deleted immediately
  - Directories will be deleted when committing the change

Making Changes – Other Commands

- `svn copy <file1> <file2>`
  - Create a new item as a copy of something else and schedule it for addition
- `svn move <file1> <file2>`
  - equivalent to "svn copy foo bar; svn delete foo"
- `svn mkdir foo`
  - equivalent to "mkdir foo; svn add foo"

Committing Changes

- svn commit –m “<log message>”
  - Sends all changes to the repository
  - Need to provide a log message with the –m option

```
[lyan1@lyan1-1 repos]$ svn delete pi.f90
D          pi.f90
[lyan1@lyan1-1 repos]$ touch void.f90
[lyan1@lyan1-1 repos]$ svn add void.f90
A          void.f90
[lyan1@lyan1-1 repos]$ svn ci -m "deleted pi.f90 and added
void.f90"
Deleting       pi.f90
Adding         void.f90
Transmitting file data .
Committed revision 4.
```

Examine Changes - Status

- `svn status`: examine the status of working copy files and directories
  - -u: add working revision and server out-of-date information
    - “*” - newer copy on the server
  - -v: display full revision information on every item
    - “?” – not under version control
    - “!” – item missing (removed by non-svn commands)

```
[lyan1@lyan1-1 repos]$ svn status -u -v
?                                       another_precis.f90
                7        1 lyan1        precis.f90
M               7        1 lyan1        laplace_solver.f90
!               7        1 lyan1        save.f90
       *        7        7 lyan1        another.f90
                7        7 lyan1        .
Status against revision:      9
```

Examine Changes - Diff

- svn diff – examine changes in detail

```
[lyan1@lyan1-1 repos]$ svn diff -r 1:4
Index: pi.f90
===================================================================
--- pi.f90      (revision 1)
+++ pi.f90      (revision 4)
@@ -1,12 +0,0 @@
-program main
-  real*8 pi8
-  real*4 pi4
-  pi8 = 3.14159265358979323846264338327950288d+0
-  pi4 = pi8
-  print *, 'PI4: ', pi4
-  print *, 'PI8: ', pi8
-  print 100, pi4
-  print 110, pi8
-100 format( e50.40 )
-110 format( d50.40 )
-end program main
Index: void.f90
===================================================================
```

CENTER FOR COMPUTATION & TECHNOLOGY

Undo Local Changes

- `svn revert <item>`
  - equivalent to deleting the item from the working copy and running `svn update`
  - Does not have to communicate with the repository to restore a file
  - Cannot restore removed directories

Resolve Conflicts (1)

- When updating
  - U and G are fine
  - C means changes from the server overlapped local changes, and decision has to be made
- For every conflicted file, the original file will be marked for conflicts
  - three un-versioned files will also be in the working copy
    - File.mine: the file as it is in the local working copy
    - File.rOLDREV: the file that was checked out before the latest edits
    - File.rNEWREV: the file received from the server
  - No commit can be made until those three files are gone

Resolving Conflicts (2)

- To resolve the conflicts, one has to
  - Merge the conflicted text "by hand"
  - Copy one of the temporary files on top of the working file
  - Run `svn revert <file>` to discard all local changes
- Need to run `svn resolved` after resolving the conflict
  - `svn revert` will automatically resolve the conflict

Resolving Conflicts (3)

```
[lyan1@lyan1-1 repos]$ svn ci -m "Edited void.f90"
Sending        void.f90
svn: Commit failed (details follow):
svn: Out of date: 'void.f90' in transaction '5-1'
[lyan1@lyan1-1 repos]$ svn up
C    void.f90
Updated to revision 5.
[lyan1@lyan1-1 repos]$ svn diff
Index: void.f90
===================================================================
--- void.f90    (revision 5)
+++ void.f90    (working copy)
@@ -1,2 +1,7 @@
+<<<<<<< .mine
+program bar
+end program
+=======
 program foo
 end program
+>>>>>>> .r5
[lyan1@lyan1-1 repos]$ ls void.f90*
void.f90  void.f90.mine  void.f90.r4  void.f90.r5
```

Exercise 2

- With your working copy
  - Change your working copy (edit a file, add a file, delete a file etc.) and commit the change
  - Try to create a conflict and resolve it
    - Check out a working copy to a different location, edit a file and commit the change
    - Edit the same part of the file and commit the change

Examining History

- Explore the history of revisions as well as the metadata
- svn log
  - Shows log messages with date and author information
- svn diff
  - Shows line-level details of a particular change
- svn cat
  - Displays any file as it exist in a particular revision
- svn list
  - Displays the files in a directory for any give revision

CENTER FOR COMPUTATION & TECHNOLOGY

Branches

- Branches are parallel to the original line of development
  - Feature-based, release-based etc.
- Create a branch using `svn copy`

Creating a Tag

- A tag is a snapshot of a project
  - Will not be changed, used to mark a milestone in the development, e.g. release
- Tags are created by using `svn copy`

Questions?

LSU
CENTER FOR COMPUTATION & TECHNOLOGY