Version Control Using Subersion

2501ICT/7421ICTNathan

René Hexel

School of Information and Communication Technology
Griffith University

Semester 1, 2012

Outline

What is Subversion?

- Version Control System
- Allows you manage the life cycle of a program
- Keep track of changes as you develop a program
- View and compare differences between versions
- Go back to an earlier version
- Create Milestones
  - Snapshot of your program at a given point in time
  - Won't change, even if your program keeps changes

How does Subversion work?

- Central repository for all versions of all your files
  - Logbook of changes
- Local working copy
  - Make changes as you go without losing information about earlier versions
- Track changes between versions
  - Make debugging easier
  - “Where did this error sneak into my program?”

An Example

- E.g. a source file `hello.m`

```
int main (void)
{
        printf("Hello, world!\n");

        return 0;
}
```

- Let's put these changes back into the repository:
  - `svn commit hello.m`
  - This is what we need to type on the command line

Preparation – required only once!

- Set up a repository on `dwarf.ict.griffith.edu.au`
  - Log into dwarf using `ssh` or putty
    - e.g. `ssh s1234567@dwarf.ict.griffith.edu.au`
  - Create the repository: `svn_setup p3`
- Log out of `dwarf`
- Create an assignment working copy on your computer (one line!):

```
svn checkout svn+ssh://s1234567@dwarf.ict.griffith.ed
u.au/export/student/s1234567/.p3svn-2012/ass1/trunk a1
```

- `cd a1`

Adding Files – required for every new file

1. Go to your checked out working directory
   - `cd a1`
2. Create a new file with your favourite editor
   - e.g. `module1.m`
3. Add the file to Subversion
   - `svn add module1.m`
4. Commit the file to the repository
   - `svn commit -m "Log Message" module1.m`
5. Repeat the last step for any changes you make to any files
   - `svn commit -m "Log Message"`
   - *Without a file name, `svn commit` will commit all files that have changed!*

Committing Changes to Subversion

- Whenever you make any changes, commit them!
  - `svn commit -m "`*`Log Message`*`"`
- Commit early, commit often!
  - Allows you more fine grained control over your changes
  - Backup copies of earlier versions
- What happens if I forget the `-m` ?
  - An editor (usually `vi`) will open
  - In `vi` you can use the `i` key to insert text: enter the log message, then press `ESC` followed by `Shift-Z Shift-Z` to save and commit.

Using Subversion on Dwarf

- So far: you used `svn` on your local machine
- Requires you to enter a password every time
  - Can be cumbersome for tagging or examining revisions
- Simply replace the remote repository URI for `dwarf`
  - `svn+ssh://sid@dwarf.ict.griffith.edu.au/export/student/sid/.p3svn-2012`
- with the local URI when logged in on `dwarf`
  - `file://$HOME/.p3svn-2012`
- Prefer a Graphical User Interface (GUI)?
  - GUI clients available for most Operating systems
    - TortoiseSVN for Windows
    - KSVN for Linux
    - MacSVN for Mac OS X

Submitting Assignments: Symbolic Tags

- The Problem:
  - Version numbers (1, 2, 3, . . .) are not very readable!
  - Every commit gets its own version number
    - . . . even if it belongs to a different project!
    - e.g. commits to Assignment 2 also changes Assignment 1
- The answer: named versions = *tags*
  - First, make sure all files are committed using `svn commit`, then run the following command on `dwarf`
  - `svn copy -m "Log"`
    `file://$HOME/.p3svn-2012/ass1/trunk`
    `file://$HOME/.p3svn-2012/ass1/tags/milestone1`
  - (all of the above needs to be on a single line!)
  - Copies the current version to a symbolic tag

Other useful Subversion Commands

- `svn log` *`[filename]`*
  - See the history of changes you made
  - Lists your log messages (make sure they are meaningful!)
  - *filename* is optional!
- `svn diff -r` *`version1:version2 [filename]`*
  - Show the actual changes between two versions
- `svn diff`
  - Show all the changes since the last `svn commit`
- `svn status` *`[filename]`*
  - Check the current version of a file

Multiple Working Copies

- What if you want multiple copies?
  - E.g., one at home, on in the labs
- Simply use `svn checkout` on multiple machines!
- Always commit all your changes after working on a program!
  - `svn commit -m "log message"`
- Bring your local copy up to date before working on any file!
  - `svn update`

Advanced Subversion Commands

- `svn update -r` *`version [filename]`*
  - go back to a specific *version*
- Update your local copy to the latest version
  - `svn update`
  - No `-r` means: go to the latest version (HEAD revision)
- `svn merge -r` *`version1`*`:`*`version2`*
  - merge the changes between two versions into the current working copy

What Else?

- There is a lot more to Subversion!
  - Branches, exporting, group work (outside of course!), etc.
- Subversion Web Page
  - `http://subversion.tigris.org/`
- Subversion Book (Online and Free!)
  - `http://svnbook.red-bean.com/`