Version Control
(Správa verzí)

http://d3s.mff.cuni.cz

Pavel Parízek
parizek@d3s.mff.cuni.cz
What is it good for?

- Keeping history of system evolution
  - Tracking progress

- Allowing concurrent work on the system
  - Teams of developers
  - Possible conflicts

- Easy reverting to a previous version
  - Safer experimentation
Typical architecture

Source code repository (versioned sources)

Working copy

synchronization

Working copy
Basic usage scenario

1. Check-out or update
2. Modify & test
3. Add & check-in

Source code repository

Working copy
Centralized versioning systems

- CVS: Concurrent Versioning System
  - The “classic” system

- SVN: Subversion
  - Currently used by many open-source projects
Subversion
Important features

- Whole source tree versioned
  - Integer numbers (1,2,3,...)
- Mixed versions in the working copy
- Atomic commits
- Versioning for files and directories
  - Operations: move, rename, delete
- Support for binary files
- Disconnected operations
- Metadata in “.svn” directories
Locations

• Repository
  ▣ Local directory (file://<absolute path>)
  ▣ Remote server (svn+ssh://<network url>)

• Working copy
  ▣ Local directory on your computer

• Always create separated local directories !!
Basic commands

- **Help:** `svn help <command>`
- **Create new repository:** `svnadmin create`
- **Create new working copy:** `svn checkout`
- **Update working copy:** `svn update`
- **List modified and new files:** `svn status -v`
- **Show differences between repository and working copy (two versions):** `svn diff -r<version>`
- **Add new files into repository:** `svn add`
- **Commit changes:** `svn commit -m "..."`
- **Display information about file:** `svn info`
Task 1

- Create repository in a local directory
  - Under $HOME/BIG
- In the working copy,
  - Create directory (e.g., “main”) where you will put everything
  - Create one file (10-20 lines) in that directory
- Add the directory and file into the repository and commit
- Create another file and commit into the repository

- Do not forget to write commit messages !!

- Commands
  - `svn checkout`, `update`, `status [-v]`, `diff`, `add`, `commit [-m]`, `info`
  - `svnadmin create`
Few more useful commands

- Undo changes in working copy: `svn revert`
- See full history of a given file: `svn log`

- Importing whole unversioned tree into repository: `svn import <dir> <repo>`
- Exporting content of the repository without metadata: `svn export`
Task 2

- Make some changes in versioned files
- Cancel them with `svn revert`
- Use `svn log` to see full history of some file
- Use `svn diff -r<v1>:<v2>` to see differences between two specific versions
Managing files and directories

- **Commands**
  - `svn add <path>`
  - `svn delete <path>`
  - `svn copy <path1> <path2>`
  - `svn move <path1> <path2>`
  - `svn mkdir <path>`

- **Path**
  - In your local working copy
  - Repository (auto-commit)
Task 3

- Try some changes in your local working copy
  - add new directory, rename file, ...
- Commit everything
- Delete the new directory in the repository
- Update your working copy
Branches and merging

Branching

Bug fix in released version

Main development

Concurrent development (experimenting)

Merging

Time & software versions

NOW
Branching and merging – commands

• Create new branch
  - `svn copy <main line repo path> <branch repo path>`

• Print differences
  - `svn diff <main line repo path> <branch repo path>`

• Make your branch up-to-date (sync merge)
  - `svn merge <main line repo path>`
  - `svn merge ^/<main line repo dir>`

• Merge branch into the main line (trunk)
  - `svn merge --reintegrate ^/<branch repo dir>`

• Preview
  - `svn merge <repo path> --dry-run`
Task 4

- Create new branch in your repository
- Checkout the branch into a new working copy
- Make some changes in the working copy for the branch, and commit immediately
- Make some changes to different files in the working copy for the main line, and commit immediately
- Print differences between the main line and branch
- Merge branch safely into the main line

Commands

- `svn copy`, `svn merge <repo path>`
- `svn merge --reintegrate`
Undoing committed modifications

- Merge negative version range into local working copy
  - `svn merge <repo path> -r <v1>:<v2>`
  - Note: v1 > v2
- Commit everything
Cherrypicking

- Merge specific change into your branch
  ```
  svn merge -c <version> <repo path>
  ```
- Commit your branch
Conflicts

- Options
  - Postpone resolving
  - Choose version
  - External merge tool
  - and many others

- Conflict markers
  - <<<<<<<<< and >>>>>>>>>> in source file

- Three variants of the source file created
Task 5

- Checkout new working copy of the main line
- Make conflicting changes to the same file in both working copies of the main line
- Commit changes in the new working copy
- Try updating the original working copy
  - It still contains uncommitted local changes
- Explore different options to resolve conflicts
Tree conflicts

- Subversion 1.6+
- Typical cause
  - Renamed files and directories
  - Deleted files
- Solution
  - Make proper changes in the working copy
  - Use patches created with `svn diff`
  - Resolve and commit
    - `svn resolve --accept=working <path>`
Task 6

- Rename some file in one working copy (WC1) of the main line, and commit

- Change this file in the other working copy (WC2)

- Update the working copy WC2
  - Tree conflict should occur now

- Solve the tree conflict properly
  - Propagate changes to the file with a new name
  - Remove the old file in the working copy WC2

- Command: `svn resolve`
Tags

- Snapshot with a human-friendly name
- Logical copy of the whole source tree
  - `svn copy <repo path 1> <repo path 2>`

- Listing all tags (directory entries)
  - `svn list <repo path>`
Standard repository layout

/trunk
/branches
/tags

/project1/trunk
/project1/branches/feature1
/project1/tags
/project2/trunk
/project2/branches
/project2/tags/R_1_0
/project2/tags/R_1_2_1
Revision keywords

- HEAD
  - Latest version in the repository

- BASE
  - Revision number in the working copy (before modifications)

- COMMITTED
  - The latest revision in which the item changed and was committed (not larger than BASE)

- PREV
  - Equal to COMMITTED-1
Best practices: synchronizing developers

• Software developed in large teams
  ▪ People may not be always able to coordinate efficiently

• Solution: Copy-Modify-Merge
  ▪ Concurrent modification of source files
  ▪ Resolving conflicts when they happen

• Alternative: Lock-Modify-Unlock
  ▪ The old classic approach ("before internet")
  ▪ Does not scale well (exclusive access)
  ▪ Not very robust (people forget to unlock)
Best practices: branches and merging

- Use branches for experimental features
- Create special branch for each feature
- Separate release and development branches
  - Propagating bugfixes from development to stable

- Merge often and synchronize with trunk
  - Lower chance of ugly conflicts occurring
  - Smaller conflicts are easier to resolve
  - Commit often ➔ others will have to merge
Properties

• Standard name-value pairs
• Many internal system properties
  ▪ svn:ignore, svn:eol-style, ...

• Setting property value
  ▪ `svn propset <name> <value> <path>`
  ▪ `svn propset <name> -F <file> <path>`

• Other commands
  ▪ `svn proplist`
  ▪ `svn propget`
  ▪ `svn propedit`
Locks

- Still needed to work with binary files
  - Merge not supported for concurrent modifications

- Locking
  - `svn lock`

- Unlocking
  - `svn commit`
  - `svn unlock`

- Inspecting
  - `svn info`
Repository access

- Local filesystem
  - UNIX permissions

- Remote
  - SSH, HTTP
GUI clients for SVN

- Tortoise SVN (Windows)
  - http://www.tortoisesvn.net

- Eclipse IDE

- Other
  - kdesvn (Linux)
  - svnx (Mac OS)
Links

- http://subversion.apache.org
- SVN Book
  - http://svnbook.red-bean.com
- Public repository servers
  - http://code.google.com
  - http://sourceforge.net
Homework

• Assignment

• Deadline
  ▪ 15.10.2014 / 20.10.2014