Fachpraktikum
Grafik-Programmierung WS2010

Introduction to Version Control

Alexandros Panagiotidis, Daniel Kauker
Institut für Visualisierung und Interaktive Systeme
Universität Stuttgart
Advantages of Version Control Systems

- Files/directories are stored in a (remote) repository
  - Snapshots of any part possible
  - Changes are logged and can be traced
  - Previous revisions can be accessed/restore
- Development can be split into multiple branches
  - Simultaneously modify branches, merge if needed
- Collaborative software development
  - Consistent view on the source code
  - Notifications if files were modified while you were working on them
  - Merging works automatically (most of the time)
How and why do we use Version Control?

- Manage assignments
  - Students retrieve program skeletons
  - Tutors retrieve solutions
- **Keep in mind:** Earlier revisions can be accessed anytime
  - Commit often! (e.g., every time something works)
  - If something stopped working ("broke"), compare with previous revisions (useful, e.g., when your program does not compile shortly before the deadline)
- Used very often in professional (software) development
  - Knowledge and practice with Version Control useful
What we use

- **Subversion (SVN)**
  - Relatively new (since 2000), open source
  - Was considered as successor to Concurrent Versioning System (CVS)
    - Nowadays more alternatives available (git, hg, VSS, p4, ...)
  - Further information
    - Project page: http://subversion.apache.org/

- **Subversion at VIS (in this course)**
  - Everyone gets his/her own course repository, located at https://svn.vis.uni-stuttgart.de/FaPras/Graphik/WS2010/<username>
  - VIS account needed for access, apply at http://www.vis.uni-stuttgart.de/ger/misc/intern/form/Rechnererlaubnis_VIS_Pool.htm
  - Accessible via command-line tool on the shell (see following slides)
    - ...also from outside the university network (no tunnel/proxy necessary)
Working with Subversion

- Preparations (we did that for you)
  1. Create repository
  2. Initial check-in

- Usual work cycle
  1. Check-out (once per working directory)
  2. Modify working copy
  3. Update
  4. Commit

- Things that can happen (not covered here)
  - Conflicts (repository changed while you were working on it)
  - Check-outs/updates do not compile
  - Manual merging necessary
  - ...

Institut für Visualisierung und Interaktive Systeme – Universität Stuttgart
Working with Subversion – Usual Work Cycle

0. Create working directory **once** (e.g., in your home directory)
   
   $ mkdir ~/fapra

1. Check-out assignment **once**
   
   $ cd ~/fapra
   $ svn co https://svn.vis.uni-stuttgart.de/FaPras/Graphik/WS2010/<username>/X
   
   (X = assignment number, i.e., 1-5)

2. Modify working copy
   
   (Use any tools you like on the working copy)

3. Update
   
   $ svn up

4. Commit
   
   $ svn ci
   
   (Enter **useful/sensible** log message)
Working with Subversion – Commands Explained

- `svn co` (checkout)
  - Copy the repository (recursively) into current local directory
  - This is your **working directory**
  - Needed once per computer you work on

- `svn up` (update)
  - Synchronizes working copy with repository
  - Merges if needed, possible conflicts may need manual resolving
  - Do this **every time** you start/stop working and **before** you commit!

- `svn ci` (commit)
  - Synchronizes repository with working copy
  - Merges if needed and possible, aborts on conflicts
Other useful commands

- `svn help` – Describe the usage of this program or its subcommands
  - Call `svn help <subcommand>` for more information/help
  - Example: `svn help co`

- `svn st` – Print the status of working copy files and directories

- `svn ls` – List the contents of the repository

- `svn info` – Display information about a local or remote item

- `svn di` – Display the differences between two revisions or paths

- `svn log` – Show the log messages for a set of revision(s) and/or file(s)

- See `svn help`, documentation, and handbook for further/additional information
The end.

Questions?