

Subversion in OS/2 and eComStation

Main principles, usage, and installation
of a free version-control system
(also) for OS/2 and eComStation

Jarda Kačer <jarda@kacer.biz>

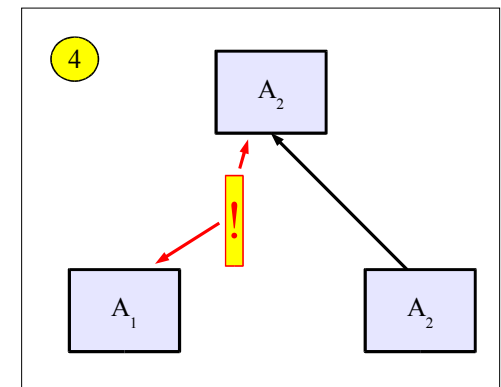
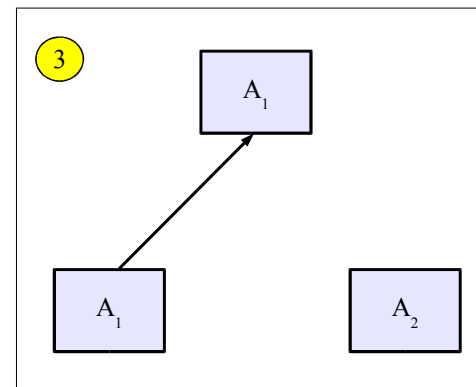
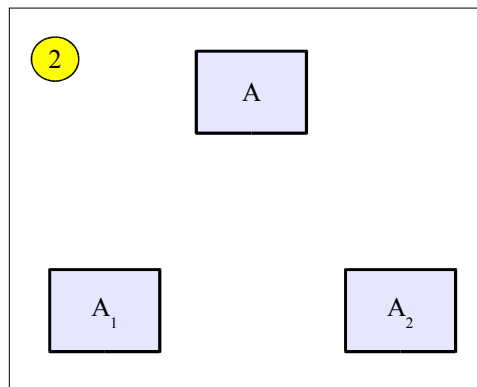
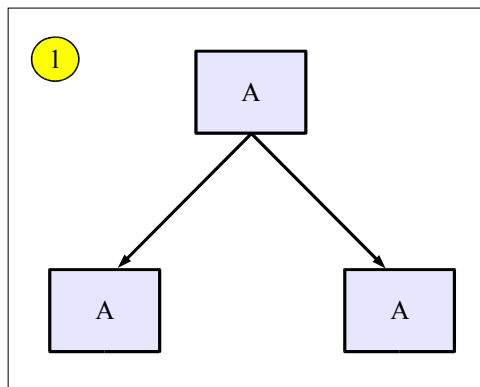
Warpstock Europe 2006
Cologne, November 16-19, 2006

Agenda

- Problems of concurrent work
- Main features and components of Subversion
- Revisions, basic principles of work
- Branching, merging, tagging, standard layout
- Installation and usage on OS/2

Problems of Concurrent Work

- One shared copy, more users working in parallel
- Classic problem known from parallel programming
- Changes made by one user are re-written with changes of another user and thus lost



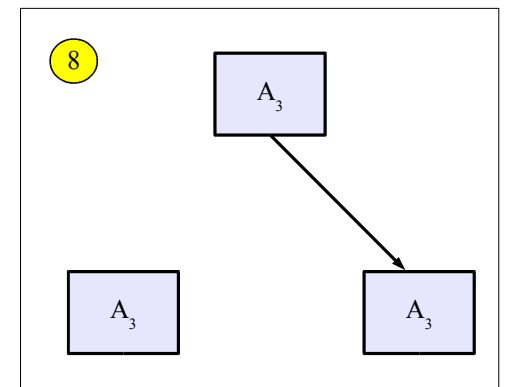
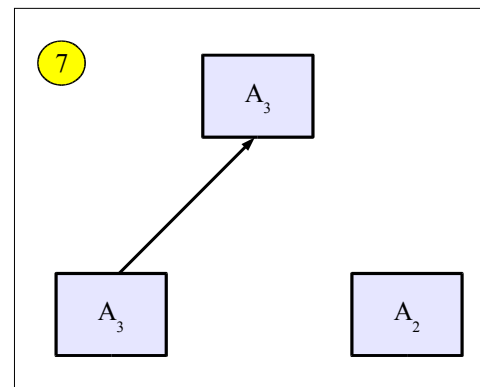
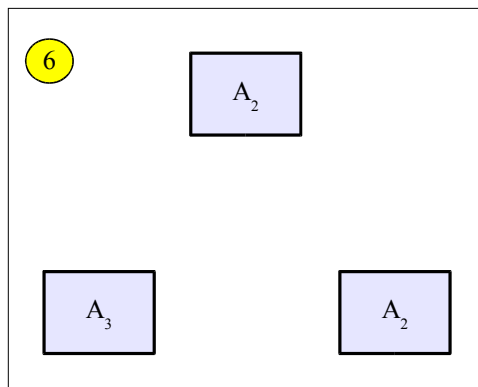
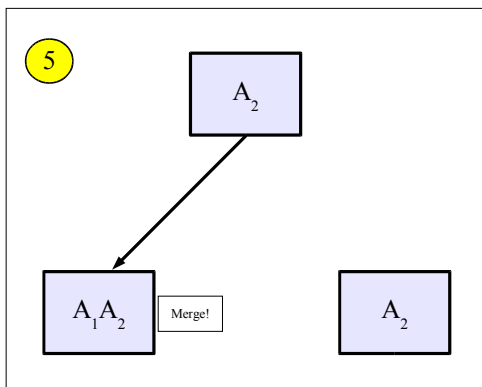
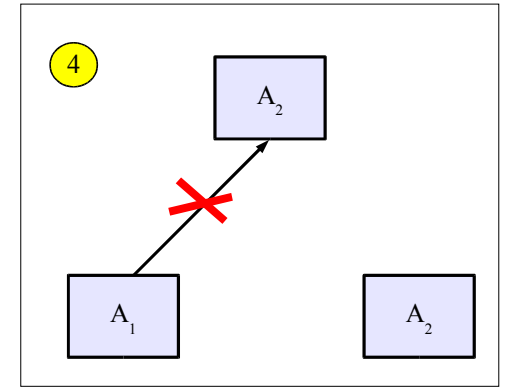
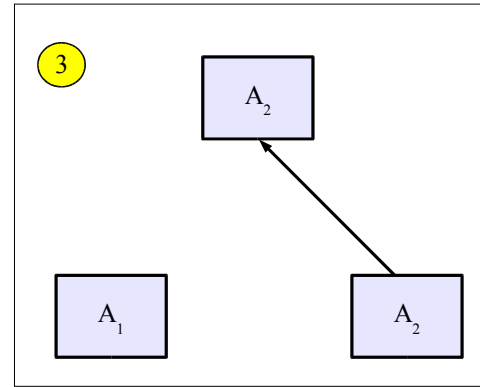
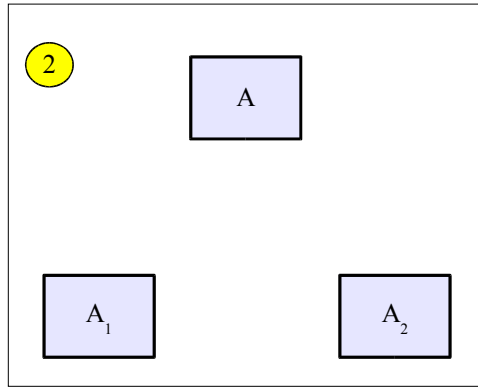
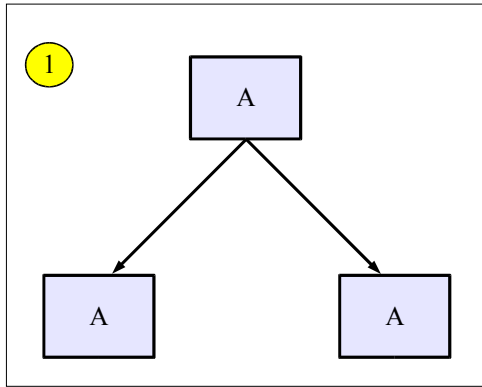
Solution 1: Locking

- A standard solution for parallel algorithms
- To be able to change the shared copy, the user must have it locked and he must have the last copy
- Not really convenient for version control – Too restrictive
 - Pessimistic algorithm
 - In reality, not exactly the same places are usually modified

Solution 2: Copy-Modify-Merge [1]

- Every user has its own **working copy** (WC)
- A shared version (for everybody) in the **repository**
- Changes performed in the WC only
- There is an algorithm to merge changes from other users to the WC
- Only a properly merged WC can be written to the repository
- A conflict can arise when merging the WC

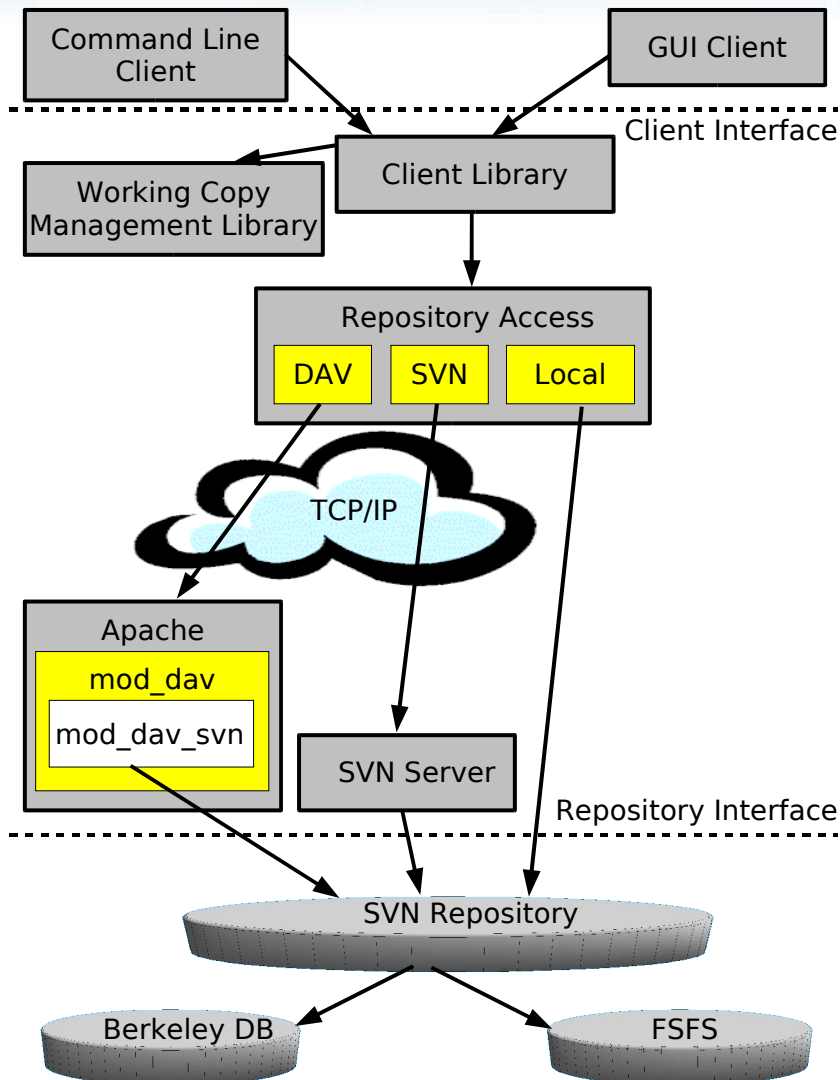
Solution 2: Copy-Modify-Merge [2]



Subversion – Basic Features

- An open-source version-control system using the Copy-Modify-Merge method
- A central repository, users' local WCs
- From the beginning planned to replace CVS
 - Able to version directories, supports file copying, moving, ...
 - Atomic commit
 - Versioned metadata
 - Different network layers
 - Efficient branching and tagging, ...

System Structure



- Different clients
- Different access methods:
 - file:// - locally
 - http:// and https://
 - Apache with WebDAV
 - svn:// and svn+ssh:// - SVN protocol
- Different types of repository

Programs in the Distribution

- `svn` – command-line client
- `svnversion` – to get WC status
- `svnlook` – to browse repository
- `svnadmin` – to configure repository
- `svndumpfilter` – to filter dump streams from repository
- `svnserv` – server
- `mod_dav_svn` – Apache module

The Most Basic Operations

- Checkout
 - `svn checkout URL project`
 - The very first copy of a project from the repository to your WC
- Update
 - `svn update`
 - To get changes made by others (from the repository) to your WC
- Commit
 - `svn commit`
 - To public your changes from your WC to the repository

Revisions

- A non-negative integer
- **Global** for the whole repository
 - For a given file F, there is not necessarily any change between revisions M and N
- Any **commit** to the repository **increments** the revision number
- A revision is a snapshot of the repository in a certain moment
- For a new repository, revision number = 0

Special Revision Keywords

- HEAD

- The newest revision in the repository

- BASE

Valid only locally in a WC

- Revision of last update of a file in the WC
- May differ for different files

- COMMITTED

- Revision of last commit of a file in the WC
- \leq BASE

- PREV

- The last-before-commit revision – COMMITTED-1

States of Files in WC [1]

- Unchanged + Current
 - No changes in WC, no changes in repository
 - Commit will not do anything
 - Update will not do anything
- Locally changed + Current
 - Changes in WC, no changes in repository
 - Commit will succeed
 - Update will not do anything

States of Files in WC [2]

- **Unchanged + Out-of-date**
 - No changes in WC, changes in repository from others
 - Commit will not do anything
 - Update will bring changes to WC
- **Locally changed + Out-of-date**
 - Changes in WC, changes in repository from others
 - Commit will fail, WC must be updated first
 - Update will try to merge changes, may succeed or fail

Commit versus Update

- Completely independent, commit does not need update, update does not need commit
 - The last slide is an exception: The commit fails because there are changes in the repository not yet propagated to the WC
- Every file can have a different revision because of commits on different files → mixed revisions
 - Normal state

Basic Work Cycle [1]

1. Update your WC

- svn update, one letter as result: U – Updated, A – Added, D – Deleted, R – Replaced, G – Merged, C – Conflict

2. Make changes

- svn add – add a new file
- svn delete – delete a file
- svn copy – make a copy
- svn move – move a file
- Or just edit some file(s)

Basic Work Cycle [2]

3. Detect changes

- svn status – info about changes
- svn diff – get the exact difference of content
- svn revert – throw away your changes, return to BASE

4. Refresh your WC

- svn update – good to do before committing (to be sure), you do not lose your local changes :-)
- svn resolved – resolve a possible conflict after update (see below)

5. Publish your changes to the repository

- svn commit – a new revision will be created

Other Useful Commands

- `svn log` – prints out history with comments
- `svn cat` – prints out a file from the repository
- `svn list` – prints out files in a dir in the repository
- `svn cleanup` – cleans up the WC after abort
- `svn import` – first import of a project into the repository
- `svn info` – info about the WC and the repository
- `svn help [command]` – help for all commands
- Usually all commands accept options in many different ways.

Conflicts During Update

- Merge cannot work because local and remote changes are at the same places
- “C” in the listing, so-called “conflict markers” inside the file: “<<<<<<<”, “=====”, “>>>>>>>”
- Moreover, 3 temporary files are created:
 - file.mine – my local file with changes, state right before update
 - file.rXXX – local file at revision BASE (without changes)
 - file.rYYY – file from the repository – HEAD

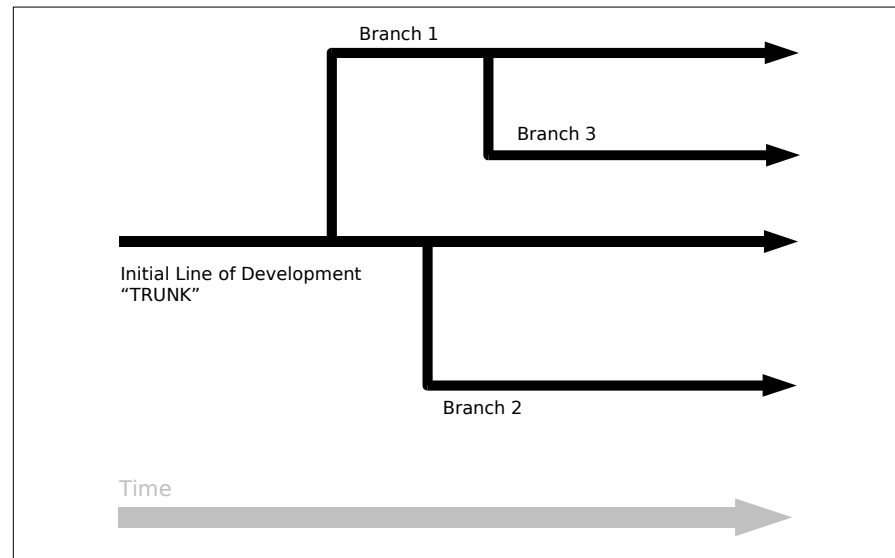
Resolution of Conflicts

- 3 options:
 - Edit by hand the problematic places between <<<<<<< and >>>>>>>
 - Use one of the temporary files and copy it over the file in question
 - svn revert – throw away your changes
- Then: svn resolved
 - Deletes all 3 temporary files and marks the file as merged so it can be committed

Project Branches [1]

- Branch:
 - Line of development independent of the others
 - It has a common history with the branch it was created from
 - Begins to exist by copying an existing branch
- Branches add one (virtual) dimension:

- File
- Revision
- Branch



Project Branching [2]

- TRUNK = Initial, main branch
- The 3rd dimension is only virtual
 - In fact, it's a directory/URL like all others
 - People give it the meaning of a branch
- `svn copy URL-trunk URL-branch -m "Description"`
- Two typical uses (see below):
 - Release branches – a branch for every released version
 - Feature branches – to prevent longer work from affecting stability of the project

Continuous Merges from Trunk to Branch

- Goal: To continuously apply changes from the trunk to our branch to prevent problems when merging the branch back to the trunk at the end (feature branches)
 - Also applies for release branches – bug fixes etc.
- So-called “porting”
- `svn merge -r Rbranch:Rtrunk http://.../trunk`
- `svn commit`
- Subversion does not remember merges done!
Be careful not to merge the same changes more than once!

Conflicts During Porting

- The file need not exist at all in your WC → Skipped missing target
- Also a “normal” conflict can occur
- Created files:
 - file.working
 - file.left
 - file.right

Porting to Trunk

- You must have the trunk in your WC, fully up-to-date
- You must know when the branch was created or last merged to the trunk
 - `svn log --verbose --stop-on-copy URL-branch`
 - `cd \trunk`
 - `svn update`
 - `svn merge -r XX:YY URL-branch`
 - `svn status`
 - `svn commit -m "Merging changes from branch B, from revisions XX-YY to trunk."`

Undo

- `svn merge -r 303:302 URL`
- `svn commit -m "Undo"`



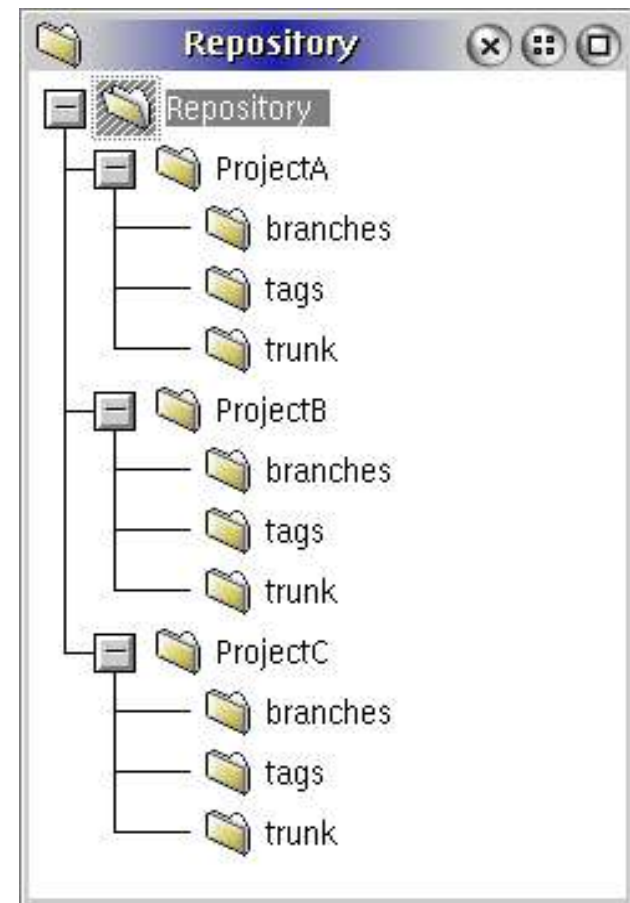
A reverse change!

Tags

- Tag = Snapshot in a certain time
- Like a revision, but has a name, not necessarily for the whole repository
- `svn copy URL/trunk URL/tags/release-1.0 -m "Tag v. 1.0"`
- Actually, it's a branch too, simply a directory
- The admin must make it read-only
- Trick: You can save your WC as a tag:
 - `svn copy MyWC URL/tags/mine`
 - If your WC is mixed from different branches, updates etc. and you want to save it

Typical Repository Layout

- Several projects in the repository
- Every project has:
 - Trunk
 - Branches
 - Tags



Software Development 1: Release Branches

- New stuff to /trunk
- Before release, copy to /branches/x.y
- Further work in parallel:
 - /trunk – development of a new version
 - /branches/x.y – testing, bug fixing
- Branch /branches/x.y is maintained
- Release to the customer: /branches/x.y → /tags/x.y.z
- Next tag of this branch will be in /tags/x.y.(z+1)

Software Development 2: Feature Branches

- A branch exists for a limited time when a new feature is developed
- Not to affect stability of /trunk
- Possible extremes:
 - Everything to /trunk, no branches
 - Nothing to /trunk directly, only to branches, porting to /trunk
- Usually somewhere in the middle:
 - Branches for longer works
 - Regularly: merge /trunk → branch
 - At the end: merge branch → /trunk

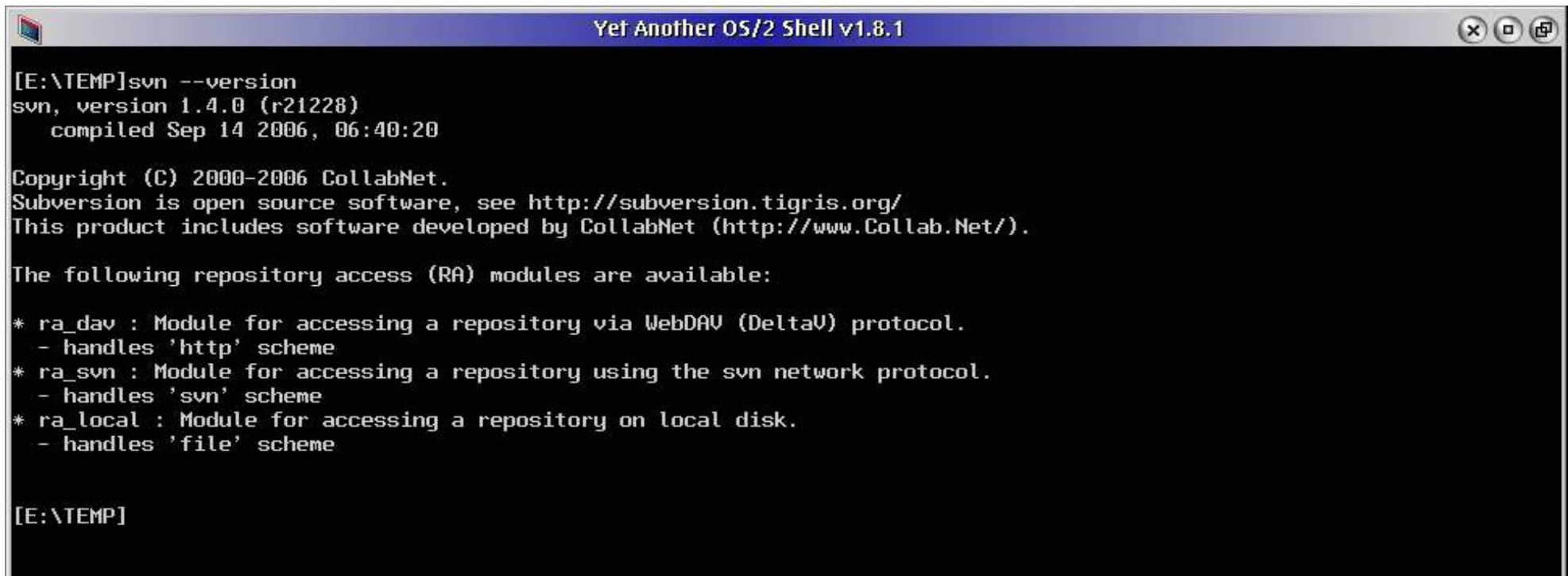
Switching between Branches

- You can switch your WC from branch to branch
- `svn switch NewURL`
- Even to a certain revision
- You can switch just directories or even files → “Mixed WC”, can become pretty messy :-)
- All branches must be from the same repository

The OS/2 Version

- Port by Paul Smedley at <http://www.smedley.info/os2ports/>
- Currently version 1.4.0 beta 1 from 2006-09-14
- Compiled with the “new” GCC, requires LibC 0.6.1
- Not much tested, even some basic operations do not work :-(
 - Bug reports: <http://mantis.smedley.info>
 - Paul does a great job but he needs some feedback!
- Installation = Unpack a ZIP file

SVN Information



```
[E:\TEMP]svn --version
svn, version 1.4.0 (r21228)
  compiled Sep 14 2006, 06:40:20

Copyright (C) 2000-2006 CollabNet.
Subversion is open source software, see http://subversion.tigris.org/
This product includes software developed by CollabNet (http://www.Collab.Net/).

The following repository access (RA) modules are available:

* ra_dav : Module for accessing a repository via WebDAV (DeltaV) protocol.
  - handles 'http' scheme
* ra_svn : Module for accessing a repository using the svn network protocol.
  - handles 'svn' scheme
* ra_local : Module for accessing a repository on local disk.
  - handles 'file' scheme

[E:\TEMP]
```

Repository Creation

```
Yet Another OS/2 Shell v1.8.1

[E:\TEMP]svnadmin create F:\Servers\SVN

[E:\TEMP]ll F:\Servers\SVN\

The volume label in drive F is SERVERS.
The Volume Serial Number is 6239:35C0.
Directory of F:\Servers\SVN

06-09-18 12:33      <DIR>      0 ---- .
06-09-18 12:33      <DIR>      0 ---- ..
06-09-18 12:33      <DIR>      0 ---- conf
06-09-18 12:33      <DIR>      0 ---- dav
06-09-18 12:33      <DIR>      0 ---- db
06-09-18 12:33          2      0 a--r format
06-09-18 12:33      <DIR>      0 ---- hooks
06-09-18 12:33      <DIR>      0 ---- locks
06-09-18 12:33      234      0 a--- README.txt
          9 file(s)      236 bytes used
          10 442 735 K bytes free

[E:\TEMP]
```

```
FC/2: F:\Servers\SVN

UP-DIR < ..
SUBDIR < conf
SUBDIR < dav
SUBDIR < db
SUBDIR < hooks
SUBDIR < locks
      2 format
     234 README.txt

FC/2: D:\SVN\Subversion\bin
UP-DIR < ..
2001490 svn.exe
422952 svnadmin.exe
369250 svndumpfilter.exe
395048 svnlook.exe
446596 svnserve.exe
1837892 svnversion.exe
```

Help

```
Yet Another OS/2 Shell v1.8.1
[E:\TEMP]svn help
usage: svn <subcommand> [options] [args]
Subversion command-line client, version 1.4.0.
Type 'svn help <subcommand>' for help on a specific subcommand.
Type 'svn --version' to see the program version and RA modules
  or 'svn --version --quiet' to see just the version number.

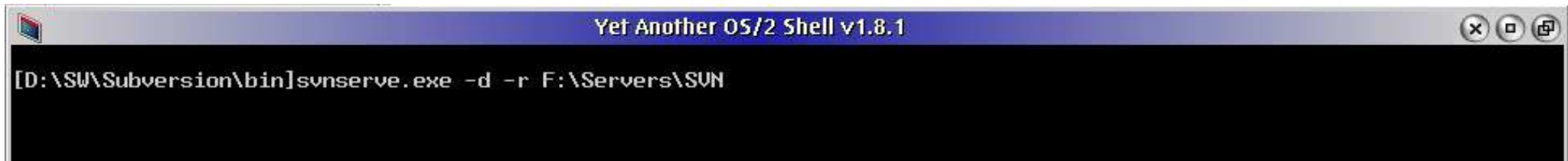
Most subcommands take file and/or directory arguments, recursing
on the directories.  If no arguments are supplied to such a
command, it recurses on the current directory (inclusive) by default.

Available subcommands:
  add
  blame (praise, annotate, ann)
  cat
  checkout (co)
  cleanup
  commit (ci)
  copy (cp)
  delete (del, remove, rm)
  diff (di)
  export
  help (?, h)
  import
  info
  list (ls)
  lock
  log
  merge
  mkdir
  move (mv, rename, ren)
  propdel (pdel, pd)
  propedit (pedit, pe)
  propget (pget, pg)
  proplist (plist, pl)
  propset (pset, ps)
  resolved
  revert
  status (stat, st)
  switch (sw)
  unlock
  update (up)

Subversion is a tool for version control.
For additional information, see http://subversion.tigris.org/

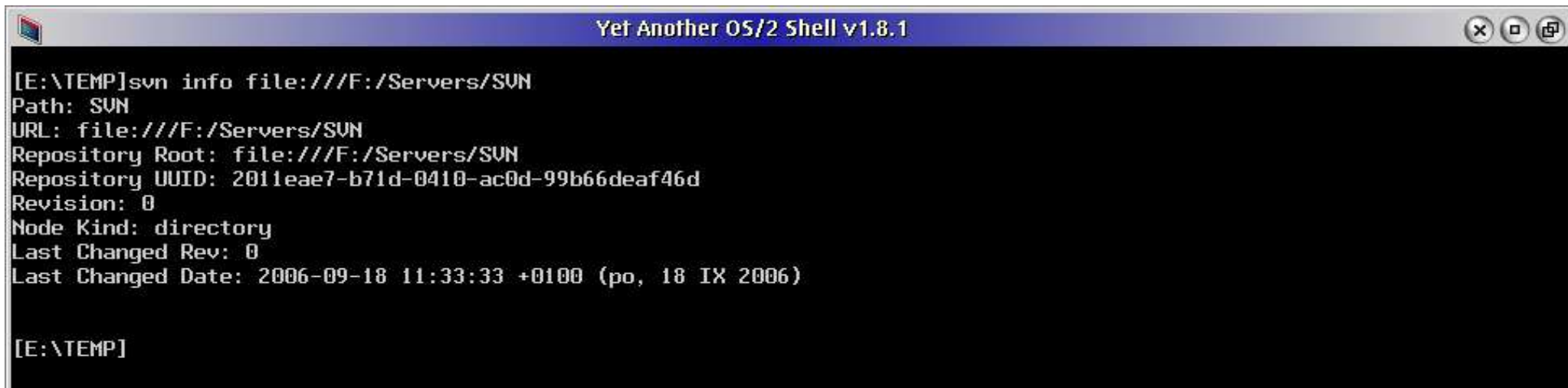
[E:\TEMP]
```

Start of the Server



```
Yet Another OS/2 Shell v1.8.1
[D:\SW\Subversion\bin]svnserve.exe -d -r F:\Servers\SUN
```

Local Access

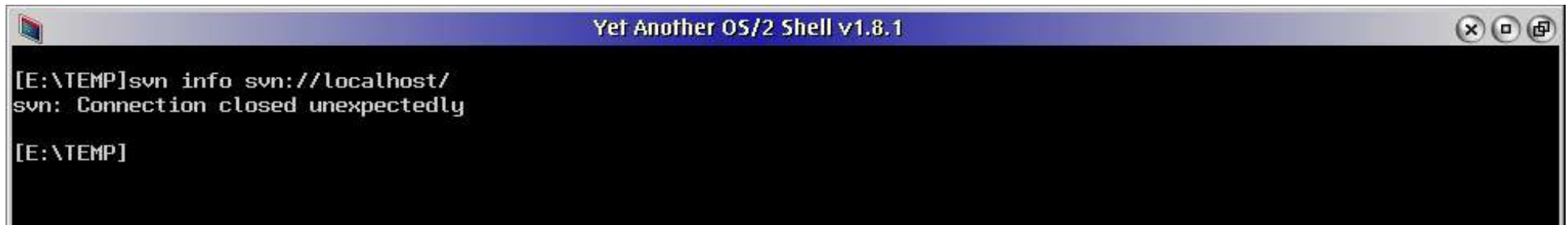


```
[E:\TEMP]svn info file:///F:/Servers/SUN
Path: SUN
URL: file:///F:/Servers/SUN
Repository Root: file:///F:/Servers/SUN
Repository UUID: 2011eae7-b71d-0410-ac0d-99b66deaf46d
Revision: 0
Node Kind: directory
Last Changed Rev: 0
Last Changed Date: 2006-09-18 11:33:33 +0100 (po, 18 IX 2006)

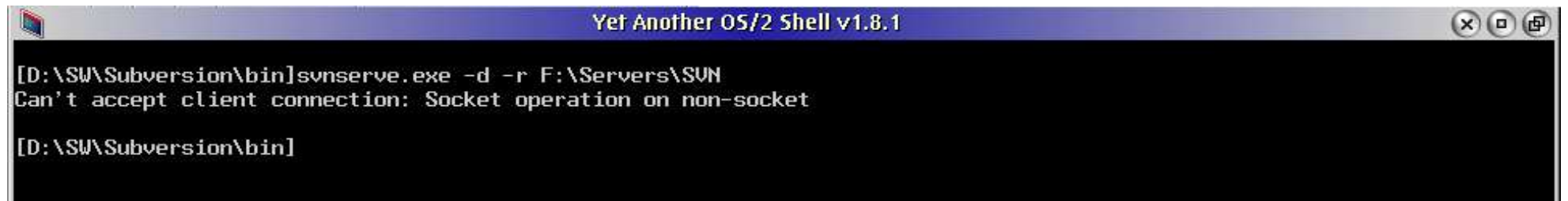
[E:\TEMP]
```

A Bug in 1.3.1

- The server crashes when a client connects :-)
- Fixed in 1.4.0 :-)

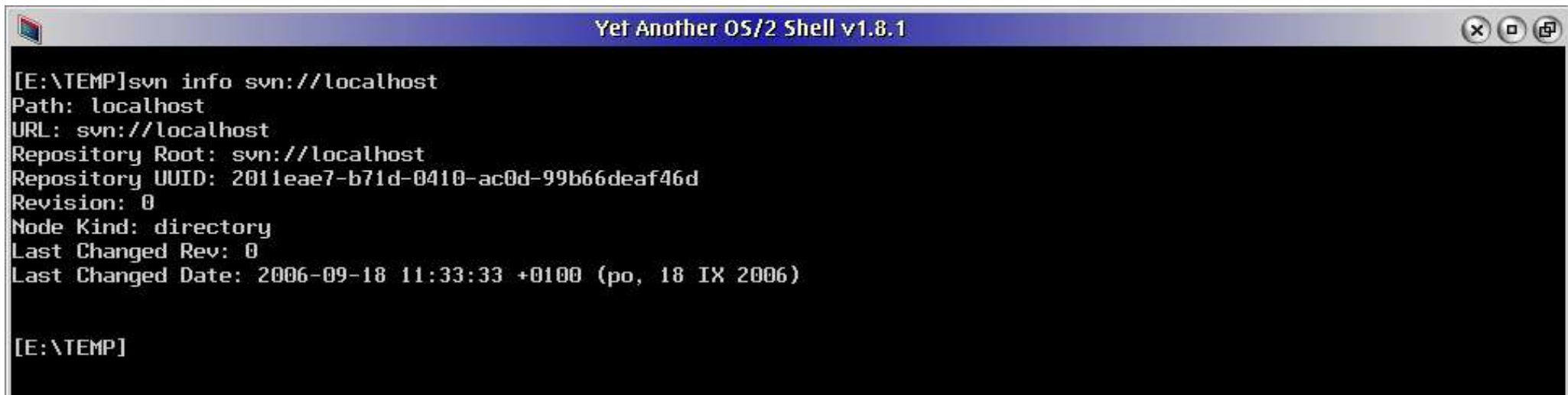


```
Yet Another OS/2 Shell v1.8.1
[E:\TEMP]svn info svn://localhost/
svn: Connection closed unexpectedly
[E:\TEMP]
```



```
Yet Another OS/2 Shell v1.8.1
[D:\SW\Subversion\bin]svnserve.exe -d -r F:\Servers\SUN
Can't accept client connection: Socket operation on non-socket
[D:\SW\Subversion\bin]
```

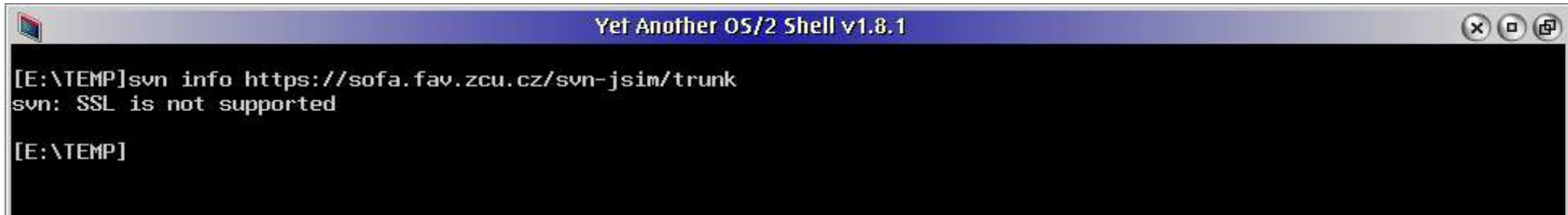
Network Access

A terminal window titled "Yet Another OS/2 Shell v1.8.1" with standard window controls (close, maximize, copy) in the top right. The terminal displays the output of the command "svn info svn://localhost".

```
[E:\TEMP]svn info svn://localhost
Path: localhost
URL: svn://localhost
Repository Root: svn://localhost
Repository UUID: 2011eae7-b71d-0410-ac0d-99b66deaf46d
Revision: 0
Node Kind: directory
Last Changed Rev: 0
Last Changed Date: 2006-09-18 11:33:33 +0100 (po, 18 IX 2006)

[E:\TEMP]
```

SSL Not Supported:-)



The image shows a terminal window titled "Yet Another OS/2 Shell v1.8.1". The terminal output displays the command `svn info https://sofa.fav.zcu.cz/svn-jsim/trunk` and the error message `svn: SSL is not supported`. The prompt `[E:\TEMP]` is visible before and after the command.

```
[E:\TEMP]svn info https://sofa.fav.zcu.cz/svn-jsim/trunk
svn: SSL is not supported
[E:\TEMP]
```

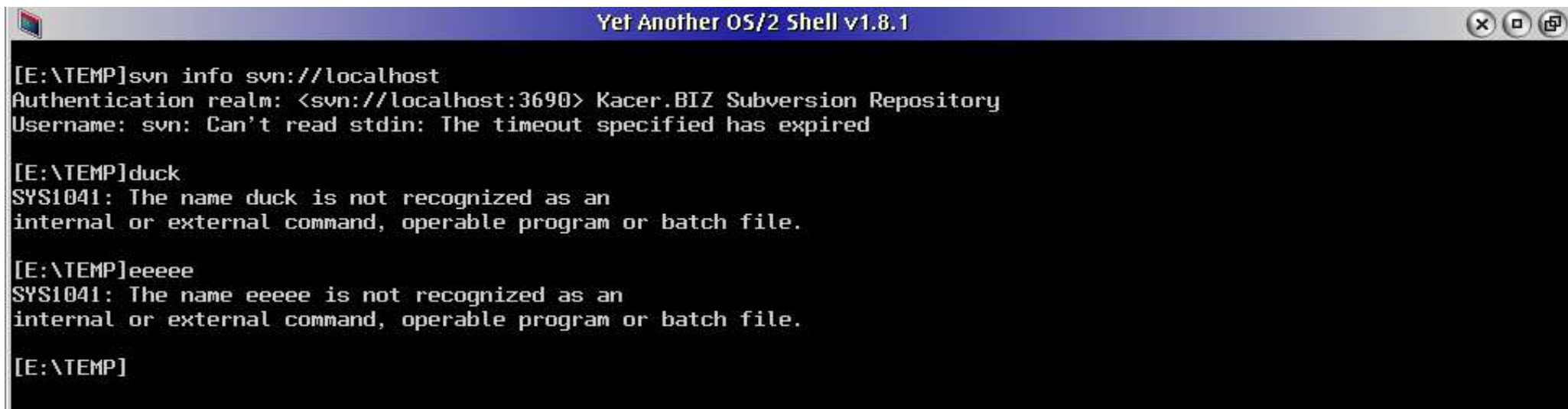

Configuration



The image shows a screenshot of a Windows Notepad window. The title bar reads "FC/2: svnserve.conf". The window content displays the configuration for the svnserve service. The configuration is as follows:

```
View: F:\Servers\SUN\conf\svnserve.conf CP852 100.00%  
[general]  
password-db = users  
realm = Kacer.BIZ Subversion Repository  
  
anon-access = none  
auth-access = write
```

Authentication Bug :-)



```
Yet Another OS/2 Shell v1.8.1
[E:\TEMP]svn info svn://localhost
Authentication realm: <svn://localhost:3690> Kacer.BIZ Subversion Repository
Username: svn: Can't read stdin: The timeout specified has expired

[E:\TEMP]duck
SYS1041: The name duck is not recognized as an
internal or external command, operable program or batch file.

[E:\TEMP]eeeeee
SYS1041: The name eeeee is not recognized as an
internal or external command, operable program or batch file.

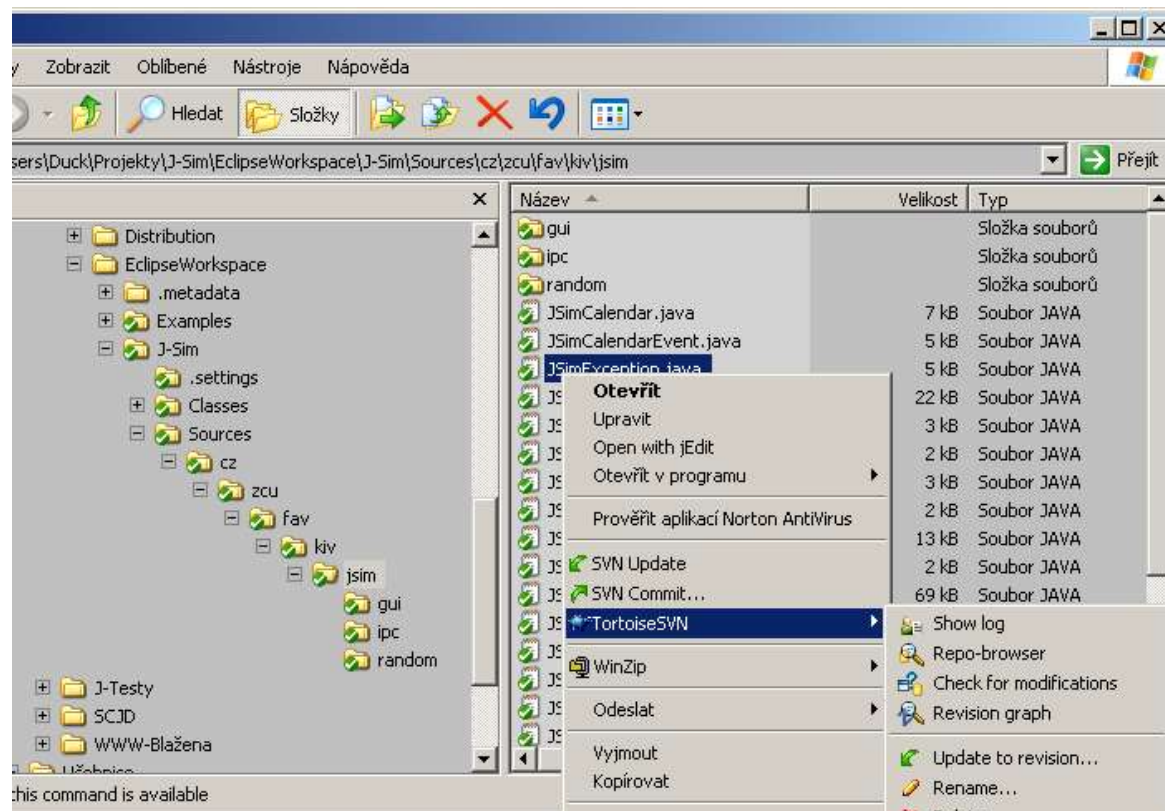
[E:\TEMP]
```

Books

- Version Control with Subversion
 - B. Collins-Sussman, B. W. Fitzpatrick, C. M. Pilato
 - <http://svnbook.red-bean.com>
- Pragmatic Version Control Using Subversion
 - M. Mason
 - <http://www.pragmaticprogrammer.com/titles/svn/>

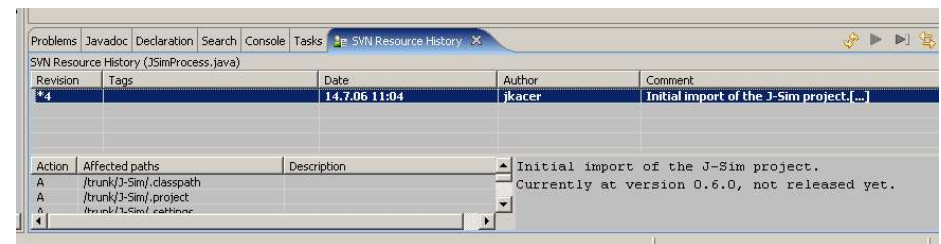
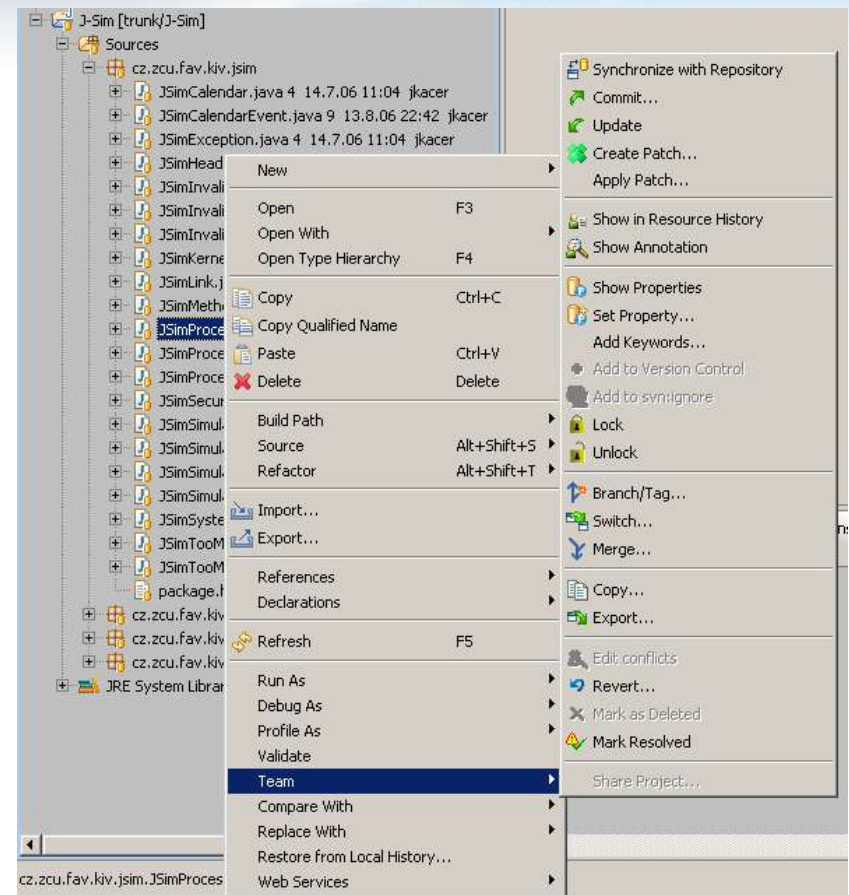
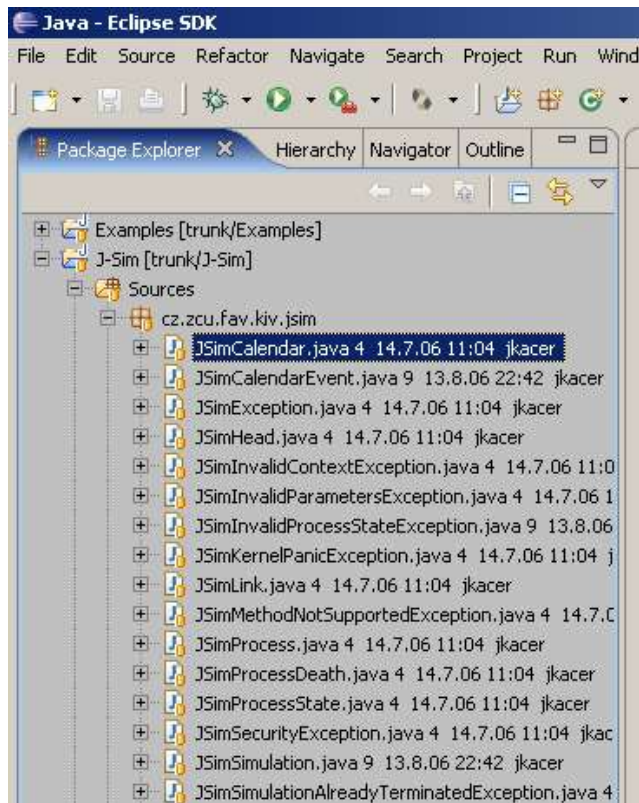
Subversion on Other Systems [1]

- Windows: Tortoise SVN
 - Explorer plugin, accessible via a context menu
 - Would also be possible in eCS as a WPS extension



Subversion on Other Systems [2]

- Eclipse: Subclipse
 - Plugin, integrated everywhere in the IDE



What We Did Not Talk About

- Repository administration
- Subversion + Apache
- Properties, hooks, access control on directory level
- Locking
- Many other topics...

Benefits of Using Version Control

- **Complete history** of your projects
- Necessary for **team development**
 - Efficient way of code sharing and change management
- You feel **safe**
 - You can rollback your changes any time
 - You do not worry so much about making changes
- It gives **stability** to your development process
 - Nothing can be lost
 - Support for older versions in parallel to the current one

Conclusion

- Excellent portable open-source version-control system
- Should replace CVS in the future
- Unfortunately, the OS/2 port is not much usable yet
 - More users necessary, as usual
 - Much work for one person – Paul Smedley