

Úvod do verzovacích systémů

(informativní)

Jan Faigl

Katedra počítačů

Fakulta elektrotechnická

České vysoké učení technické v Praze

Přednáška 12

A0B36PR2 – Programování 2

Přehled témat

Základní pojmy verzování souborů

CVS - Concurrent Version System

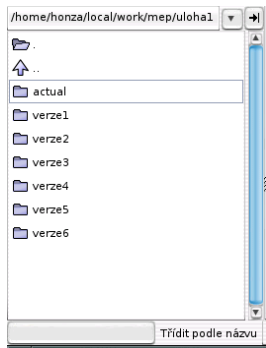
SVN - Subversion – vybrané pokročilé vlastnosti

Distribuované verzovací systémy – základní filozofie

Verzování

Proč používat správce verzí

- Zálohování starších verzí pro „jistotu”.
- Vyzkoušení nového směru bez ztráty původní verze.
- Jak distribuovat soubory při více členném týmu?



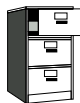
Základní pojmy

- Správce verzí - má na starosti automatické číslování verzí.
Sada nástrojů pro přístup k verzovaným souborům.
- Repozitář (Repository) - místo, kde jsou uloženy verzované soubory.
- Lokální kopie (Working copy) - lokální kopie repositáře, resp. konkrétní verze souborů z repositáře.
 - Uživatel pracuje s kopiemi verzovaných souborů, které modifikuje.

U konkrétních systémů verzování můžeme dále rozlišit lokální a pracovní kopii. Například subversion udržuje v pracovní kopii ještě adresář `.svn`, ve kterém je lokální kopie konkrétní verze repositáře, na které uživatel pracuje. Git má lokální verzi repositáře v adresáři `.git`.

Základní schema

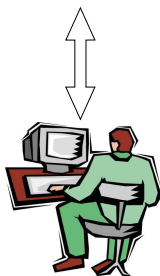
Správce verzí



Repositář



Lokální kopie



Lokální kopie

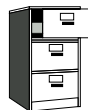


Lokální kopie

Získání lokální kopie - checkout

- Vytvoření lokální kopie verzovaných souborů.
- Adresářová struktura většinou obsahuje pomocné soubory s informacemi o verzi souborů.
- Při změnách modifikujeme lokální kopii příslušné verze.

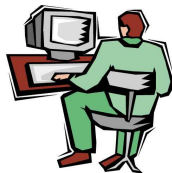
Správce verzí



Repositář



checkout

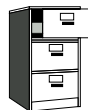


Lokální kopie

Potvrzení změn - commit

- Žádost o přijetí lokálních modifikací jako nové verze.
- Správce verzí vytvoří nejbližší vyšší verzi.
- V repositáři není novější verze, než lokální kopie, jinak:
 - Aktualizace lokální kopie na aktuální novou verzi.
 - Řešení konfliktů.
- Je slušné změny komentovat.

Správce verzí



Repositář



commit

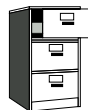


Lokální kopie

Aktualizace - update

- Aktualizace lokální kopie na novou verzi.
- Pokud jsou změny verzovaných souborů v souladu s lokálními modifikacemi probíhá sloučení změn (merge).
- Jinak řešení konfliktů.

Správce verzí



Repositář



update



Lokální kopie

Řešení konfliktů

- Správce verzí nezabraňuje vzniků konfliktů, ale nabízí nástroje pro jejich řešení.
- Konflikt vzniká převážně současnou modifikací stejného místa v souboru.
- Konfliktům lze zabránit vhodným rozčleněním projektů na moduly a případně vyhrazení práv modifikací jednotlivým vývojářům.

Sloučený soubor s konfliktem

```

1169     fprintf(stdout, "%d [%.3lf, %.3lf]\n", i,
1170     }
1171 <<<<<< vis.cpp
1172     G=12*cities.number;
1173     //G=12.41*4+0.06;
1174 =====
1175     G=12.41*cities.number+0.06;
1176 >>>>>> 1.12.2.48
1177     separate = false;
1178     return 0;
1179 }
1180
1181 //-----
1182 int CMap::coords_size(double * min_x, double * m
1183 {

```

Vizualizace rozdílů

The screenshot shows a diff tool window titled "vis.cpp (CVS r1.12.2.3) vs vis.cpp (CVS r1.12.2.28) - TKDiff 4.0". The interface includes a menu bar (File, Edit, View, Mark, Merge), a toolbar with icons for Merge, Diff, and Mark, and a status bar at the bottom showing "1 of 166".

The diff view is split into two panes. The left pane shows the code from "vis.cpp (CVS r1.12.2.3)" and the right pane shows the code from "vis.cpp (CVS r1.12.2.28)".

Line 4227 in the left pane is highlighted in grey, indicating a deletion in the newer version. The corresponding line in the right pane is highlighted in blue, indicating an addition. The diff tool uses a color scheme where grey highlights deletions and blue highlights additions.

```
4219 }
4220 //-----
4221 int CVisiblePolygon::convert_to_gpc(void)
4222 {
4223     // fprintf(stdout, "konvert to gpc %d\n", number_point
4224     // gpc_polygon gpc1;
4225     // gpc = new gpc_polygon;
4226     gpc_vertex_list contour;
4227     gpc_free_polygon(&visible_gpc_polygon);
4228
4229     gpc_vertex * body = new gpc_vertex[number_points];
4230
4231     for (int i = 0; i < number_points; i++)
4232     {
4233         body[i].x = _points[i].x; body[i].y = _points[i].y;
4234         // fprintf(stdout, "%lf %lf\n", body[i].x, body[i].
4235     }
4236     contour.vertex = body;
4237     contour.num_vertices = number_points;
4238     gpc_add_contour(&visible_gpc_polygon, &contour, 1);
4239
4240     return 0;
4241 }
4242
4243 //-----
4244 //gpc_polygon * gpc(void);
4245 gpc_polygon * CVisiblePolygon::gpc(gpc_polygon * gpc)
```

```
5237 }
5238 //-----
5239 int CVisiblePolygon::convert_to_gpc(void)
5240 {
5241     // fprintf(stdout, "konvert to gpc %d\n", number_point
5242     // gpc_polygon gpc1;
5243     // gpc = new gpc_polygon;
5244     gpc_vertex_list contour;
5245     gpc_free_polygon(&visible_gpc_polygon);
5246
5247     gpc_vertex * body = new gpc_vertex[number_points];
5248
5249     for (int i = 0; i < number_points; i++)
5250     {
5251         body[i].x = _points[i].x; body[i].y = _points[i].y;
5252         // fprintf(stdout, "%lf %lf\n", body[i].x, body[i].
5253     }
5254     contour.vertex = body;
5255     contour.num_vertices = number_points;
5256     gpc_add_contour(&visible_gpc_polygon, &contour, 1);
5257     delete[] body;
5258     return 0;
5259 }
5260
5261 //-----
5262 //void CVisiblePolygon::print(void)
5263 {
5264     for (int i = 0; i < number_points; i++)
5265         fprintf(stdout, "id %d %3lf %3lf\n", i, _points[i]
5266 }
5267 //gpc_polygon * gpc(void);
5272 gpc_polygon * CVisiblePolygon::gpc(gpc_polygon * gpc)
```

Značkování verzí - tag

- Správce verzí zachycuje historii vývoje jednotlivých souborů.
- Konkrétní stav repositáře, lze označit (tag) např. Release_1.0.
- Tag - symbolické jméno pro konkrétní verzi.
- Pro aktuální verzi se používá značka HEAD.

Větve - branch

Umožňují:

- Paralelní vývoj.
- Postupný přechod na novější technologie.
- Odzkoušení nových přístupů.

Používané větve:

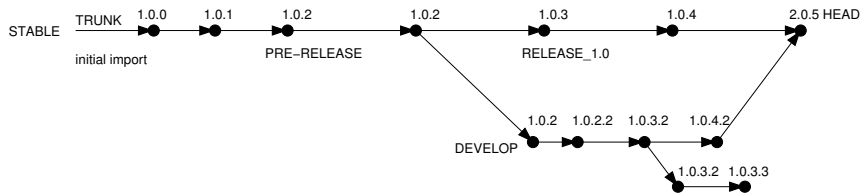
- CURRENT, TRUNK - hlavní vývojová větev.
- STABLE - stabilní vývojová větev.

Commit do STABLE větve by neměl narušit činnost ostatních vývojářů.

- Hlavní vývojová větev může být zároveň stable větví.

S výhodou se používá slučování větví (branch merge).

Příklad více větví



CVS - Concurrent Version System

- Jeden z první rozšířených systémů verzování
Pro lokální verzování např. souborů v /etc lze použít ještě jednodušší systém RCS
- Každý soubor je číslován zvlášť.
- Verze lze označit (tag), existují dynamické a statické tagy.
- Rozlišuje mezi adresářem a souborem.
- Ze serveru jsou posílány změnové soubory (diff), na server jsou však uploadovány celé soubory.
- Implicitně pracuje s textovými soubory, binární soubory je nutné označit.
- Náročnější slučování větví.
- Obtížný přesun adresářů či přejmenování souborů.

Trocha historie

Verzovací systém Subversion

- <http://subversion.apache.org>
- <http://svnbook.red-bean.com>
- Aktuální poslední verze 1.9.0-rc1 Released - 11. května, 2015
- Historie:
 - Milestone 1 - září 2000,
 - Subversion 0.8 - leden 2002,
 - Subversion 0.37 (1.0.0-RC1) - leden 2004,
 - Subversion 1.0.0 - únor 2004,
 - Subversion 1.1.0 - září 2004,
 - Subversion 1.2.0 - květen 2005,
 - Subversion 1.3.0 - leden 2006,
 - Subversion 1.4.0 - září 2006,
 - Subversion 1.5.0 - červen 2008,
 - Subversion 1.6.0 - březen 2009
 - Subversion 1.7.0 - říjen 2011 (Apache Foundation),
 - Subversion 1.8.0 - červen 2013,

SVN - Přehled základní vlastností

- Nerozlišuje mezi souborem a adresářem.

V porovnání s CVS velký pokrok.

- Více možností volby přístupu
(*ssh, svnserver, http a https - apache2 mod_dav_svn module*).

- Pro ukládání používá nativní souborový systém nebo Berkeley Data Base (db)

- “Netaguje”, ale vytváří větve

Používá levné kopie souborů/adresářů.

- Jednodušší slučování větví.
- Snadný přesun souborů.
- Na uživatelské úrovni nerozlišuje textové a binární soubory.
- **Číslování vždy celého repositáře.**

SVN - základní příkazy

- `svnadmin repos` - vytvoření repositáře *repos*.
- `svn checkout path_to_repos` - získání lokální kopie.
- `svn commit` - v adresářové struktuře lokální kopie, potvrzení změn aktuálního adresáře.
- `svn commit path` - potvrzení změn souboru/adresáře *path*.
- `svn update` - aktualizace aktuálního adresáře lokální kopie.
- `svn resolved` - označení vyřešeného konfliktu.
- `svn status` - vypsání stavu souborů.
- `svn diff` - výpis změn lokálních modifikací oproti lokální kopii.
- `svn log` - výpis zpráv.
- `svn help` - nápověda.

Levné kopie

Systém Subversion využívá mechanismu takzvaných levných kopií, kterým efektivně v repositáři ukládá shodné soubory.

- Příkazem `svn copy` vytvoříme kopii verzovaného souboru.
- V lokálním adresáři se vytvoří kopie souboru.
- V repositáři je však uložen pouze záznam o novém souboru (adresáři), který je založen na konkrétní verzi již verzovaného souboru.

Mechanismus levných kopií je využit pro

- tagy - explicitní označení repositáře v čase,

např. release_1

- větve - alternativní vývojové větve projektu.

např. devel, trunk, stable

SVN - pokročilé vlastnosti

- properties - vlastnosti verzovaných souborů,
- tags - značky,
- branches - větve a slučování větví,
- hooks - automatizace operací při interakci s repositářem.

Properties

- Každý soubor/adresář může mít několik verzovaných metadat, takzvaných vlastností (properties).
- *Property* je dvojice jméno-hodnota:
 - jméno je textový řetězec (ASCII),
 - data mohou být libovolná, podobně jako verzované soubory.
- Jména i obsah *property* je verzován.
- Vyhrazená jména systému Subversion jsou uvozeny `svn:`.
- Použití a význam ostatních vlastností je na uživateli.
- Základní příkazy pro práci s vlastnostmi jsou:
 - `svn propset` - nastavení,
 - `svn propdel` - zrušení,
 - `svn propget` - zobrazení,
 - `svn propedit` - editace,
 - `svn proplist` - výpis.

Revision properties

- Vlastnosti vztahující se ke konkrétní revizi.
- Tyto vlastnosti nejsou verzované.
- Vlastnosti jsou nastaveny při vytváření nové revize.
- Mezi tyto vlastnosti patří například:
 - `svn:author` - autor revize,
 - `svn:date` - čas vytvoření revize,
 - `svn:log` - zpráva popisující revizi.

Některé verzované systémové vlastnosti

- **svn:ignore** - seznam souborů, které mají být ignorovány příkazem `svn status`.
- **svn:eol-style** - nastavení konce řádků.
- **svn:needs-lock** - soubor je v pracovní kopii označen pouze pro čtení. Před editací musí být uzamčen.
- **svn:keywords** - nahrazení vyhrazených slov ve verzovaném souboru.
- **svn:externals** - obsahuje seznam URL, externí repositáře.

svn:keywords - nahrazování proměnných

Některá jména proměnných:

- *Date*
- *Revision*
- *Author*
- *HeadURL*
- *Id*

Příklad

```

Janra@claxton:~/work/predmety/pte2008/slides/exanples/svn/src
At revision 1.
claxton$ cat main.cc
/*
 * File name: main.cc
 * Date:    2009/01/10 19:03
 * Author:  Jan Faigl
 */

#include <iostream>

int main(int argc, char* argv[] {
    std::cout << "$Date$" << std::endl;
    return EXIT_SUCCESS;
}

/* end of main.cc */
claxton$ g++ main.cc: ./a.out
$Date$
claxton$ svn up
U    main.cc
Updated to revision 2.
claxton$ g++ main.cc: ./a.out
$Date: 2009-01-10 19:06:53 +0100 (Sat, 10 Jan 2009) $
claxton$ █
  
```

Příklad nastavení

- 1 `svn propset svn:keywords "Date" main.cc`
- 2 `svn commit -m "set svn:keywords"`

svn:externals - dokaz do jiného repositáře

- Kombinace zdrojů z více repositářů.
- Každý řádek definuje jeden adresář a jeho zdroj.

Příklad - knihovny třetích stran

```

honz@claxton:~/work/predmety/pte/2008/slides/examples/svn/externals
claxton$ ls
build.xml lib src
claxton$ svn propset svn:externals lib
log4j svn+ssh://localhost/repositories/vendors/log4j/current
exerces svn+ssh://localhost/repositories/vendors/xerces/current

claxton$ svn st
X lib/log4j
X lib/exerces

Performing status on external item at 'lib/log4j'

Performing status on external item at 'lib/exerces'
claxton$ █
  
```

Vlastnosti vs Log zprávy

- Vlastnosti lze s výhodou využít pro jednoznačné nastavení konkrétní proměnné.
- Systémové vlastnosti (proměnné) obsahují údaje, které není nutné psát do zpráv při *commitu*.
- Přes výše uvedené může být vhodné informace uvést i ve zprávě, neboť vlastnosti se relativně obtížně prohlédávají.
- Formát log zprávy lze předepsat pro snadnější parsování
log-message templating

Log zprávy

- Obsah zprávy by měl vystihovat podstatu změny.
- Rozsáhlé změny by měly být dostatečně komentovány.
- Zprávy mají také charakter komunikace mezi vývojáři.
- Zprávy lze mimo jiné využít pro vytváření souboru *ChangeLog*.
- Většina příkazů svn má možnost výstupu do xml včetně svn log.

Příklad XML výstupu

```
$ svn -r 62:62 log --xml
<?xml version="1.0"?>
<log>
  <logentry
    revision="62">
      <author>kordam2</author>
      <date>2008-12-10T19:59:36.941186Z</date>
      <msg>mala uprava.</msg>
    </logentry>
</log>
```

Tags - značky - 1/2

- Explicitní označení stavu repositáře v čase.
- Realizovány kopíí adresáře (mechanismus levných kopíí).

Příklad - značka *version1*

```

1 $ svn info
2 Path: .
3 URL: https://comrob/svn/pte1430/
4 Revision: 1
5 Node Kind: directory
6 Last Changed Rev: 1
7 $ ls
8 dijkstra
9 $ ls dijkstra
10 build.xml src
11 $ ls dijkstra/src/
12 Dijkstra.java
13 $ svn mkdir -m "Create tags directory" \
14     https://comrob/svn/pte1430/tags
15 Committed revision 2.
16 $ svn copy -m "Create tag of Dijsktra" \
17     https://comrob/svn/pte1430/dijkstra \
18     https://comrob/svn/pte1430/tags/version1
19 Committed revision 3.
```

Tags - značky - 2/2

Příklad - značka *version1* - aktualizace pracovního adresáře 2/2

```
1 $ svn up
2 A   tags
3 A   tags/version1
4 A   tags/version1/src
5 A   tags/version1/src/Dijkstra.java
6 A   tags/version1/build.xml
7 Updated to revision 3.
8 # výpis adresářové struktury
9 $ svn ls --depth infinity
10 dijkstra/
11 dijkstra/build.xml
12 dijkstra/src/
13 dijkstra/src/Dijkstra.java
14 tags/
15 tags/version1/
16 tags/version1/build.xml
17 tags/version1/src/
18 tags/version1/src/Dijkstra.java
```

Adresářová struktura

- Větvě i značky jsou adresáře.
- V repositáři může být více projektů.
- Projekt je složen z více modulů, každý modul může mít několik vývojových větví.

Dva základní přístupy:

- Větvě (značky) pro celý projekt.
- Větvě (značky) pro každý modul.

Příklad adresářové struktury

Celý projekt

```

/dijkstra
/dijkstra/trunk
/dijkstra/branches
/dijkstra/tags
/dijkstra/trunk/src
/dijkstra/branches/linear
/dijkstra/branches/heap_naive
/dijkstra/tags/submission
/dijkstra/tags/static_arrays
/dijkstra/branches/linear/src
.
.
.
.
.
.
.

```

Každý modul

```

/common
/gui
/readlog
/polygon_filter
/common/trunk
/common/branches
/common/tags
/gui/trunk
/gui/branches
/gui/tags
/readlog/trunk
/readlog/branches
/readlog/tags
.
.
.

```

Adresářová struktura - příklad I. 1/6

Historie revizí:

1. vytvoření adresářové struktury,
2. první verze souborů,
3. nová větev pro lineární prohledávání

```
$ svn copy dijkstra/trunk dijkstra/branches/linear
$ svn commit
```

4. implementace načítání v trunk

```
$ svn ci -m "Implement load_graph"
Adding          dijkstra/trunk/src/load_graph.c
Adding          dijkstra/trunk/src/load_graph.h
Transmitting file data ..
Committed revision 4.
```

5. nová větev pro haldu

```
$ svn copy dijkstra/trunk dijkstra/branches/heap
A          dijkstra/branches/heap
$ svn ci -m "Create heap branch"
Adding          dijkstra/branches/heap

Committed revision 5.
```


Adresářová struktura - příklad I. pokračování 2/6

6. implementace lineárního vyhledávání,

```
$ svn add dijkstra/branches/linear/src/linear.?
A      dijkstra/branches/linear/src/linear.c
A      dijkstra/branches/linear/src/linear.h
$ svn ci -m "Implementation of linear search"
Adding      dijkstra/branches/linear/src/linear.c
Adding      dijkstra/branches/linear/src/linear.h
Transmitting file data ..
Committed revision 6.
```

7. přidání lineárního vyhledávání do trunk,

```
$ svn merge dijkstra/trunk@3 dijkstra/branches/linear@6
      dijkstra/trunk
--- Merging differences between repository URLs into '
      dijkstra/trunk':
A      dijkstra/trunk/src/linear.h
A      dijkstra/trunk/src/linear.c
$ vim dijkstra/trunk/src/dijkstra.c
$ svn ci -m "Use linear search"
Sending      dijkstra/trunk
Sending      dijkstra/trunk/src/dijkstra.c
Adding      dijkstra/trunk/src/linear.c
Adding      dijkstra/trunk/src/linear.h
Transmitting file data .
Committed revision 7.
```

Adresářová struktura - příklad I. pokračování 3/6

8. implementace haldy,

```
$ svn ci -m "Heap has been implemented and tested"
Adding          dijkstra/branches/heap/src/heap.c
Adding          dijkstra/branches/heap/src/heap.h
Transmitting file data ..
Committed revision 8.
```

9. použití haldy v trunk,

```
$ svn merge dijkstra/trunk@5 dijkstra/branches/heap@8
dijkstra/trunk
--- Merging differences between repository URLs into '
dijkstra/trunk':
A    dijkstra/trunk/src/heap.h
A    dijkstra/trunk/src/heap.c

$ vim dijkstra/trunk/src/dijkstra.c

$ svn ci -m "Replace linear search by heap"
Sending        dijkstra/trunk
Sending        dijkstra/trunk/src/dijkstra.c
Adding        dijkstra/trunk/src/heap.c
Adding        dijkstra/trunk/src/heap.h
Transmitting file data .
Committed revision 9.
```

Adresářová struktura - příklad I. pokračování 4/6

10. označení odevzdávaného trunku,

```
$ svn copy dijkstra/trunk dijkstra/tags/submission
A      dijkstra/tags/submission
$ svn propset status TOREVIEW dijkstra
$ svn ci -m "dijkstra has been finished"
Sending      dijkstra
Adding      dijkstra/tags/submission
Adding      dijkstra/tags/submission/src
Adding      dijkstra/tags/submission/src/dijkstra.c
Adding      dijkstra/tags/submission/src/heap.c
Adding      dijkstra/tags/submission/src/heap.h

Committed revision 10.
```

Adresářová struktura - příklad I. pokračování 5/6

Přepínání mezi větvemi.

```

$ svn info
Path: .
URL: https://comrob/svn/pte1430/dijkstra
Repository Root: https://comrob/svn/pte1430
Revision: 10
Last Changed Rev: 10

$ svn switch https://comrob/svn/pte1430/dijkstra/trunk

D   trunk                A   src/heap.h
D   branches             A   src/main.c
D   tags                 A   src/load_graph.h
A   src                  A   src/linear.c
A   src/linear.h         A   src/dijkstra.c
A   src/dijkstra.h       A   src/heap.c
A   src/load_graph.c
U   .
Updated to revision 10

$ svn info
Path: .
URL: https://comrob/svn/pte1430/dijkstra/trunk
Repository Root: https://comrob/svn/pte1430
Revision: 10
Node Kind: directory
Last Changed Rev: 9

```

Adresářová struktura - příklad I. pokračování 6/6

Přepnutí na větev heap a zpět na trunk.

```
$ svn switch https://comrob/svn/pte1430/dijkstra/branches/heap
D src/linear.h
D src/linear.c
U src/dijkstra.c
U .
Updated to revision 10
```

```
$ svn info
Path: .
URL: https://comrob/svn/pte1430/dijkstra/branches/heap
Repository Root: https://comrob/svn/pte1430
Revision: 10
Last Changed Rev: 8
```

```
$ svn switch https://comrob/svn/pte1430/dijkstra/trunk
A src/linear.h
A src/linear.c
U src/dijkstra.c
U .
Updated to revision 10.
```

```
$ svn info
Path: .
URL: https://comrob/svn/pte1430/dijkstra/trunk
Repository Root: https://comrob/svn/pte1430
Revision: 10
Last Changed Rev: 9
```

Adresářová struktura - příklad II. 1/4

Jak vytvořit lokální adresářovou strukturu pro sestavení projektu z více modulů?

```

common/
common/branches/
common/trunk/
common/trunk/errors.h
common/trunk/logging.h
gui/
gui/branches/
gui/trunk/
gui/trunk/cairoPainter.c
gui/trunk/cairoPainter.h
gui/trunk/gui_window.c
gui/trunk/gui_window.h
readlog/
readlog/branches/
readlog/trunk/
readlog/trunk/laser_scan.c
readlog/trunk/laser_scan.h
$ svn co https://comrob/svn/sensors
A sensors/gui
A sensors/gui/trunk
A sensors/gui/trunk/cairoPainter.c
A sensors/gui/trunk/gui_window.h
A sensors/gui/trunk/cairoPainter.h
A sensors/gui/branches
A sensors/common
A sensors/common/trunk
A sensors/common/trunk/errors.h
A sensors/common/trunk/logging.h
A sensors/common/branches
A sensors/readlog
A sensors/readlog/trunk
A sensors/readlog/trunk/laser_scan.h
A sensors/readlog/trunk/laser_scan.c
A sensors/readlog/branches
Checked out revision 2.
```

Adresářová struktura - příklad II. pokračování 2/4

Přepnutí jednotlivých modulů na trunk.

```
$ cd sensors
```

```
$ svn switch https://comrob/svn/sensors/gui/trunk gui
```

```
D   gui/trunk
```

```
D   gui/branches
```

```
A   gui/cairoPainter.c
```

```
A   gui/gui_window.h
```

```
A   gui/cairoPainter.h
```

```
A   gui/gui_window.c
```

```
Updated to revision 2.
```

```
$ svn switch https://comrob/svn/sensors/common/trunk common
```

```
D   common/trunk
```

```
D   common/branches
```

```
A   common/errors.h
```

```
A   common/logging.h
```

```
Updated to revision 2.
```

```
$ svn switch https://comrob/svn/sensors/readlog/trunk readlog
```

```
D   readlog/trunk
```

```
D   readlog/branches
```

```
A   readlog/laser_scan.h
```

```
A   readlog/laser_scan.c
```

```
Updated to revision 2.
```

Adresářová struktura - příklad II. pokračování 3/4

```
$ svn info
Path: .
URL: https://comrob/svn/sensors
Repository Root: https://comrob/svn/sensors
Revision: 2
Last Changed Rev: 2
```

```
$ svn update
At revision 4
```

```
$ svn -v log -l 2
```

```
-----
r4 | standa | 2007-10-14 01:10:36 (Sat, 14 Oct 2007) | 1 line
```

```
Changed paths:
```

```
A /gui/branches/glPainter/glPainter.c
```

```
A /gui/branches/glPainter/glPainter.h
```

```
M /gui/branches/glPainter/guiWindow.c
```

```
implemented gl painter
```

```
-----
r3 | standa | 2007-10-13 18:11:13 (Fri, 13 Oct 2007) | 1 line
```

```
Changed paths:
```

```
A /gui/branches/glPainter (from /gui/trunk:2)
```

```
Create glPainter branch
```


Adresářová struktura - příklad II. pokračování 4/4

```
$ svn switch https://comrob/svn/sensors/gui/branches/glPainter gui
A   gui/glPainter.c
A   gui/glPainter.h
U   gui/guiWindow.c
Updated to revision 4.
```

```
$ svn info common
Path: common
URL: https://comrob/svn/sensors/common/trunk
Repository Root: https://comrob/svn/sensors
Revision: 4
Last Changed Rev: 2
```

```
$ svn info gui
Path: gui
URL: https://comrob/svn/sensors/gui/branches/glPainter
Repository Root: https://comrob/svn/sensors
Revision: 4
Last Changed Rev: 4
```

```
$ svn info readlog/
Path: readlog
URL: https://comrob/svn/sensors/readlog/trunk
Repository Root: https://comrob/svn/sensors
Revision: 4
Last Changed Rev: 2
```

Adresářová struktura - příklad III. 1/5

- Samostatný repositář pro knihovny třetích stran.

```
$ svn --depth infinity ls https://comrob/svn/vendors
log4j/
log4j/1.2.13/
log4j/1.2.13/log4j.jar
log4j/1.2.9/
log4j/1.2.9/log4j.jar
log4j/current/
log4j/current/log4j.jar
xerces/
xerces/2.6.2/
xerces/2.6.2/xercesImpl.jar
xerces/2.6.2/xml-apis.jar
xerces/2.7.1/
xerces/2.7.1/xercesImpl.jar
xerces/2.7.1/xml-apis.jar
xerces/current/
xerces/current/xercesImpl.jar
xerces/current/xml-apis.jar
```

- V projektu nastavíme property `svn:externals`.

Adresářová struktura - příklad III. pokračování 2/5

- Aplikace pro načítání xml.

```
$ svn --depth infinity ls https://comrob/svn/pte1430
task2_xml/
task2_xml/trunk/
task2_xml/trunk/libs/
```

- V projektu používáme knihovny *log4j* a *xerces*, v *trunk* budeme používat verze 1.2.9 a 2.6.2.

```
log4j https://comrob/svn/vendors/log4j/1.2.9
exercer https://comrob/svn/vendors/xerces/2.6.2
```

- Nastavíme property *svn:externals*.

```
svn propedit svn:externals https://comrob/svn/pte1430/task2_xml/
trunk/libs -m "Set vendors libs"
Set new value for property 'svn:externals' on
'https://comrob/svn/pte1430/task2_xml/trunk/libs'

Committed revision 2.
```

Adresářová struktura - příklad III. pokračování 3/5

■ Provedeme *checkout*.

```
$ svn checkout https://comrob/svn/pte1430
```

```
A    pte1430/task2_xml
```

```
A    pte1430/task2_xml/trunk
```

```
A    pte1430/task2_xml/trunk/libs
```

```
Fetching external item into 'pte1430/task2_xml/trunk/libs/log4j'
```

```
A    pte1430/task2_xml/trunk/libs/log4j/log4j.jar
```

```
Checked out external at revision 4.
```

```
Fetching external item into 'pte1430/task2_xml/trunk/libs/exerces'
```

```
A    pte1430/task2_xml/trunk/libs/exerces/xml-apis.jar
```

```
A    pte1430/task2_xml/trunk/libs/exerces/xercesImpl.jar
```

```
Checked out external at revision 4.
```

```
Checked out revision 2.
```

Adresářová struktura - příklad III. pokračování 4/5

- Přejít na nové verze knihoven provedeme v samostatné větvi.

```
$ svn mkdir https://comrob/svn/pte1430/task2_xml/branches -m "  
Create branch directory"
```

Committed revision 3.

```
$ svn copy https://comrob/svn/pte1430/task2_xml/trunk https://  
comrob/svn/pte1430/task2_xml/branches/new_vendor_libs/ -m "  
Create branch for new libs"
```

Committed revision 4.

- Použijeme verze 1.2.13 a 2.7.1.

```
#log4j https://comrob/svn/vendors/log4j/1.2.13  
#exercis https://comrob/svn/vendors/xerces/2.7.1
```

```
$ svn propedit svn:externals https://comrob/svn/pte1430/task2_xml/  
branches/new_vendor_libs/lib
```

Adresářová struktura - příklad III. pokračování 5/5

- Přepneme na novou větev.

```
svn switch https://comrob/svn/pte1430/task2_xml/
branches/new_vendor_libs pte1430/task2_xml/
D pte1430/task2_xml/trunk
A pte1430/task2_xml/libs
```

```
Fetching external item into 'pte1430/task2_xml/libs/log4j'
A pte1430/task2_xml/libs/log4j/log4j.jar
Updated external to revision 4.
```

```
Fetching external item into 'pte1430/task2_xml/libs/exerces'
A pte1430/task2_xml/libs/exerces/xml-apis.jar
A pte1430/task2_xml/libs/exerces/xercesImpl.jar
Updated external to revision 4.
```

Updated to revision 5.

```
$svn info pte1430/task2_xml/libs/exerces/
Path: pte1430/task2_xml/libs/exerces
URL: https://comrob/svn/vendors/xerces/2.7.1
Repository Root: https://comrob/svn/vendors
Revision: 4
Last Changed Rev: 3
```

Hooks

- Při interakci s repositářem je možné definovat skripty, které provedou příslušnou operaci.
- Adresářová struktura repositáře (na serveru) obsahuje adresář `hooks`.
- Pro konkrétní událost je volán skript s konkrétním jménem: např. `post-commit`, `pre-commit`.
- Skript je volán s definovanými argumenty, např. cesta k repositáři a číslo revize.
- Pro přístup k repositáři lze použít program `svnlook`.
- Po vytvoření repositáře, např. příkazem `svnadmin`, obsahuje adresář šablony pro jednotlivé skripty (`hooks`).

Příklad - post-commit hook

```
#!/bin/sh
REPOSITORY="$1"
REV="$2"

SVNLOOK=/usr/local/bin/svnlook

CHANGES='`$SVNLOOK changed $REPOSITORY | grep -e'^_U[[:space:]]+ uloha
[[:digit:]]/$`'
STUDENTS_MAILS='`$SVNLOOK pg $REPOSITORY mails / | tr `\\n` ` ` `'

for i in $CHANGES
do
    STATUS='`$SVNLOOK pg $REPOSITORY status $i`'
    case $STATUS in
        TOREVIEW )
            #send mail
            ;;
        CHECKED)
            if [ -n "$STUDENTS_MAILS" ]
            #mails are not set
            then
                RESULT='`$SVNLOOK log $REPOSITORY`'
                #send mail with $RESULT
            fi
            ;;
    esac
done
```


Distribuované verzovací systémy – (DVCS)

- Nemají centrálně uložený repositář.
- Každý vývojář si v podstatě udržuje vlastní *lokální* repositář.
- Velmi časté používání vývojových větví.
- Výsledná verze je vlastně sestavením příslušných vývojových větví jednotlivých vývojářů.
- Příklad existujících DVCS:
 - BitKeeper – *proprietary software*
 - Bazaar – **bzr**
 - Perforce – *proprietary software*
 - Git – **git**
 - Darcs – **darcs**
 - Mercurial - **hg**
 - Plastic SCM – *proprietary software*

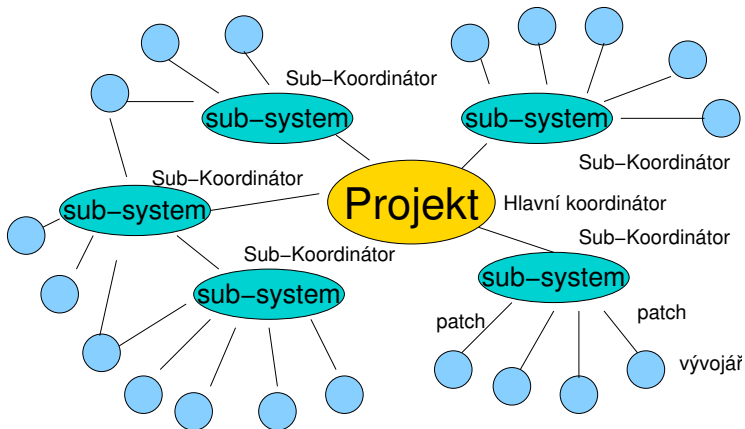
Vlastnosti

- Můžeme verzovat i bez připojení k síti.
- Centrální repositář je často nahrazen zodpovědným vývojářem.
- Vyžadují hlubší porozumění struktuře projektu.
- Vhodnost nasazení záleží na povaze projektu (modelu vývoje).

Některé distribuované verzovací systémy

- BitKeeper - <http://www.bitkeeper.com>.
- darcs - David's Advanced Revision Control System - napsaný v haskell <http://darcs.net>.
- Monotone - <http://monotone.ca>.
- Mercurial - <http://www.selenic.com/mercurial/wiki>
- SVK - založeno na subversion filesystem knihovně <http://svk.bestpractical.com/view/HomePage>.
- **git** - vytvořen pro potřeby vývoje jádra Linux <http://git-scm.com>

Model vývoje s velkým počtem vývojářů



GIT - základní vlastnosti

- Lokální repositář.
- Efektivní pro rozsáhlé projekty.
- Soubory jsou uloženy jako objekty v databázi (INDEX).
- SHA1 otisk souboru slouží jako identifikátor souboru.
- Low-level operace nad databázemi jsou zapouzdřeny uživatelsky přívětivějším rozhraním.
- Výrazná podpora pro vývojové větve.
- Podpora pro aplikování patch setů, např. z mailu.
- Základní příkazy <http://git-scm.com/documentation>
- Git - SVN Crash Course
<http://git-scm.com/course/svn.html>

Základní použití

- Vytvoření kopie repositáře pro udržování vlastních změn; `git clone`.
- Sledování jiných repositářů; `git remote`, `git fetch`.
- Vlastní vývoj a lokální verzování; `git add`, `git status`, `git log`, `git merge`, `git branch`, `git checkout`.
- Publikování změn do jiného (vzdáleného) repositáře; `git push`.

GIT - základní příkazy

<code>git init</code>	<code>svnadmin create repo</code>
<code>git clone url</code>	<code>svn checkout url</code>
<code>git add file</code>	<code>svn add file</code>
<code>git commit -a</code>	<code>svn commit</code>
<code>git pull</code>	<code>svn update</code>
<code>git status</code>	<code>svn status</code>
<code>git log</code>	<code>svn log</code>
<code>git rm <i>file</i></code>	<code>svn rm <i>file</i></code>
<code>git mv <i>file</i></code>	<code>svn mv <i>file</i></code>
<code>git tag -a <i>name</i></code>	<code>svn copy repo/trunk repo/tags/<i>name</i></code>
<code>git branch <i>branch</i></code>	<code>svn copy repo/trunk repo/branches/<i>branch</i></code>
<code>git checkout <i>branch</i></code>	<code>svn switch repo/branches/<i>branch</i></code>

Co všechno verzovat?

- Verzování zdrojových kódů programů.
- Verzování knihoven třetích stran.
- Verzování dokumentů (text/binární).
 - File and Directory Layout for Storing a Scientific Paper in Subversion

<http://blog.plesslweb.ch/post/6628076310/file-and-directory-layout-for-storing-a-scientific>

- Verzování jako „zálohování”.

Repositář je na serveru zpravidla uložena na zálohovaném diskovém systému.
- Verzování jako prostředek sdílení.

Užívat rozumně!