Mercurial v příkladech

v0.36

napsal

Thorbjörn Jemander

přeložil

ing. Jaroslav Kubias

(tovim @ seznam.cz)

© 2009-2010 Thorbjörn Jemander. Všechna práva vyhrazena.

Žádná část této práce nesmí být použita pro vytvoření odvozeného díla; jeho elektronické či papírové kopie nesmí být prodávány či jiným způsobem komerčně využívány bez předchozího písemného souhlasu autora.

Použití bez souhlasu je omezeno na čtení a kopírování pouze pro osobní potřebu.

Názory čtenářů jsou vítány. Kontaktní informaci lze nalézt na www.jemander.se

1. Obsah

1 Úvod

Napadlo mne, že by nebyl od věci návod k Mercurialu založený na příkladech, bez množství teorie a popisů zákulisních vztahů. Zde jej předkládám, bez dlouhého řečnění..

Neobávejte se, že je příliš dlouhý, projdete jím poměrně rychle.

1.1 Klíč ke čtení textu

Následující tabulka popisuje formátování textu, které se v dokumentu používá.

Rámeček	Popis

`$ ls --flag # this command`
`foo.txt`
`bar.txt # this file`
	Příkazy terminálu, $ představuje výzvu. Vstup je **tučný,** výstup není tučný.
	`Červený text` označuje zajímavý vstup, jádro příkladu.
	`Modrý text` označuje zajímavý výstup.
	`#Green text` je použit pro komentáře.

Tip: if you write...	Poznámka, tip, odkaz na další četbu

`Foo`
`bar`
Soubor Myfile.sh
	Obsah souboru, v tomto případě Myfile.sh

Není radno přeskakovat příklady, protože pozdější příklady mohou záviset na výsledcích předcházejících příkladů.

1.2 Poznámka překladatele

Téměř všechna dokumentace o Mercurialu má jednu nevídanou zvláštnost. Jeden z jejích klíčových pojmů, slovo *repozitář*, nemá pevný význam. Jednou je tímto slovem označován systémový adresář . *hg,* jindy se tímto slovem označuje kořenový adresář složky . *hg,* do třetice se toto slovo používá pro označení úložného prostoru obecně. Této mnohoznačnosti se nevyhnul ani překládaný text.

Duálním označením je dvojice slov *revize / changeset*. I když je mezi nimi jemný významový rozdíl, označují prakticky totéž.

Dalším magickým pojmem v dokumentaci Mercurialu je sousloví *pracovní adresář*. Je to kořenový adresář složky .hg. Pracovním souborům (a složkám) tohoto adresáře se říká *pracovní kopie*. Skladba a obsah souborů pracovní kopie se mění podle aktuální revize. Implicitně je aktuální poslední revize. Tuto lokalizaci lze změnit. Souborům pracovní kopie se říká *lokální soubor*, nebo *lokální kopie*. Sousloví *lokální kopie* se ale také může vztahovat na repozitář.

Nakonec chci zdůraznit, že *Mercurial v příkladech* není systematickým učebním textem, ale výtečným uceleným souborem příkladů. Bohužel, ne všechny lze věrně replikovat v OS Windows.

Ještě poznamenám, že slovem *kód* se míní text programu, nebo obecně zpracovávaný text.

2 Instalace Mercurialu

Používáte-li Debian-Linux PC, který má správce apt, můžete provést následující:

```
$ sudo apt-get install mercurial
$ hg version
Mercurial Distributed SCM (version 1.3.1)

Copyright (C) 2005-2009 Matt Mackall <mpm@selenic.com> and others
This is free software; see the source for copying conditions. There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
```

Pro jiné operační systémy viz http://mercurial.selenic.com/wiki/Download.

3 Základy

3.1 Vytvoření repozitáře

Vytvoření repozitáře Mercurialu v prázdném adresáři s použitím příkazu `init` :

```
$ mkdir hg1
$ cd hg1

$ hg init
```

Vytvořili jsme lokálně repozitář v adresáři *hg1*. Repozitář (*viz pozn. 1.2*) obsahuje jak *pracovní kopii*, tak *data repozitáře* (záznam historie). Toto se liší od centralizovaných systémů, jako je CVS nebo SVN, kde repozitář obsahuje pouze historii revizí a pracovní kopie obsahuje pouze vybraný kód.

3.2 Přidání, odebrání a zápis souboru

Všechny změny, které budeme chtít sledovat, musime repozitáři procedurálně předat. Předávací procedura se nazývá `commit.` Změny, které chceme předat, musíme předtím vhodně označit.

Vytvořme si nějakou historii:

```
$ echo A > x1

$ hg add x1
$ hg ci -m"Added feature A to module x1."    # zkrácený příkaz hg commit
```

Toto je velmi podobné SVN. Vytvořme další historii a naučme se ji odstranit:

```
$ echo A > x2
$ echo B > y1

$ hg add x2 y1
$ hg ci -m"Added A to x2 and B to y1."
$ hg rm x1                                # zkratka od hg remove
$ hg ci -m"Removed x1"                    # záznam změny
```

Nyní jsme přidali a odebrali soubory explicitně. Existuje však příkaz `addremove` který přidané a odebrané soubory označí za nás automaticky sám:

```
$ echo C > z1
$ rm x2                    # Nikoliv "hg rm"!

$ hg addremove
removing x2
adding z1
$ hg status
A z1
R x2
$ hg ci -m"Adding C to z1 and removing x2"
```

Status je příkaz ke zjištění stavu pracovní kopie před komitem. Řekne nám mimo jiné, zda jsou soubory (A)dded, (R)emoved, (M)odified. Vyzkoušejte také příkaz **manifest**.

Příkaz hg status nám poskytne užitečnou informaci ještě před komitem.

Pohleďme nyní na náš záznam:

```
$ hg log
changeset:   3:c9d133e1c054
tag:         tip
user:        Thorbjorn Jemander
date:        Sat Dec 12 10:15:49 2009 +0100
summary:     Adding C to z1 and removing x2

changeset:   2:3d8afaf1ac30
user:        Thorbjorn Jemander
date:        Sat Dec 12 10:13:09 2009 +0100
summary:     Removed x1

changeset:   1:01e9f6a4aae5
user:        Thorbjorn Jemander
date:        Sat Dec 12 10:12:09 2009 +0100
summary:     Added A to x2 and B to y1.

changeset:   0:593b00eec7b1
user:        Thorbjorn Jemander
date:        Sat Dec 12 09:49:22 2009 +0100
summary:     Added feature A to module x1
```

Vidíme přímočarý průběh, počínající revizí 0 a končící revizí 3. Poslední revize je automaticky opatřena názvem *tip*.

Chceme-li, můžeme si zobrazit jenom část historie určením rozsahu revizí:

```
$ hg log -r 1:2          # Od 1 do 2
changeset:   2:3d8afaf1ac30
user:        Thorbjorn Jemander
date:        Sat Dec 12 10:13:09 2009 +0100
summary:     Removed x1

changeset:   1:01e9f6a4aae5
user:        Thorbjorn Jemander
date:        Sat Dec 12 10:12:09 2009 +0100
summary:     Added A to x2 and B to y1.
$ hg log -r :1           # Až k 1 včetně.
changeset:   1:01e9f6a4aae5
user:        Thorbjorn Jemander
date:        Sat Dec 12 10:12:09 2009 +0100
summary:     Added A to x2 and B to y1.

changeset:   0:593b00eec7b1
user:        Thorbjorn Jemander
date:        Sat Dec 12 09:49:22 2009 +0100
summary:     Added feature A to module x1
$ hg log -r 2:           # Od 2 včetně.
changeset:   3:c9d133e1c054
tag:         tip
user:        Thorbjorn Jemander
date:        Sat Dec 12 10:15:49 2009 +0100
summary:     Adding C to z1 and removing x2

changeset:   2:3d8afaf1ac30
user:        Thorbjorn Jemander
date:        Sat Dec 12 10:13:09 2009 +0100
summary:     Removed x1
```

4 Historie

4.1 Navigace a tagy

Než se posuneme dál – kdepak jsme právě nyní? Dozvíme se to příkazem `identify`:

```
$ hg identify -n
3:15f1571783e3 tip
```

Nacházíme se v revizi 3. Pro pohyb v historii repozitáře používáme příkaz `update` :

```
$ hg update -r 0                                    # Přesunout k revizi 0.
1 files updated, 0 files merged, 2 files removed, 0 files unresolved
$ ls
x1
```

Použití `-r` je zpravidla závazné, ale u příkazu `update` lze označení `-r` vypustit. Většinu příkazů lze také zkrátit:

```
$ hg up                        # Revize neurčena → přejdeme k poslední.
1 files updated, 0 files merged, 2 files removed, 0 files unresolved
$ ls
y1 z1
```

Bez argumentu u příkazu `update` se dostaneme k revizi s označením *tip*. Jak bylo již dříve zmíněno, *tip* je symbolický název poslední revize. Můžeme sami přiřadit vlastní symbolické názvy jednotlivým revizím. Těmto názvům říkáme *tagy*:

```
$ hg tag -r 2 my-x1-removal     # Přiřazení názvu k revizi 2
$ hg log
changeset:   4:4207f3b7f4ee
tag:         tip
user:        Thorbjorn Jemander
date:        Sat Dec 12 16:41:03 2009 +0100
summary:     Added tag my-x1-removal for changeset 3d8afaf1ac30

changeset:   3:c9d133e1c054
user:        Thorbjorn Jemander
date:        Sat Dec 12 10:15:49 2009 +0100
summary:     Adding C to z1 and removing x2

changeset:   2:3d8afaf1ac30
tag:         my-x1-removal
user:        Thorbjorn Jemander
date:        Sat Dec 12 10:13:09 2009 +0100
summary:     Removed x1

changeset:   1:01e9f6a4aae5
user:        Thorbjorn Jemander
date:        Sat Dec 12 10:12:09 2009 +0100
summary:     Added A to x2 and B to y1.
```

```
changeset:   0:593b00eec7b1
user:        Thorbjorn Jemander
date:        Sat Dec 12 09:49:22 2009 +0100
summary:     Added feature A to module x1
$ hg tags
tip                                4:4207f3b7f4ee
my-x1-removal                      2:3d8afaf1ac30
```

Výpis použitých tagů zajistí příkaz `hg tags`, tagy se také objevují v záznamu revizí. Jména tagů lze použít ve většině příkazů místo čísel revizí.

4.2 Příkazy diff a cat

Soubory a adresáře lze přejmenovat:

```
$ hg rename y1 y2                      # událost
$ hg ci -m"y1 renamed to y2"           # její záznam do repozitáře

$ ls
y2  z1
$ echo C > y2

$ hg ci -m"Updated module y2 with feature C."
```

Nový obsah, nové jméno a přesto je identita souboru sledována. Všimněme si, že potřebujeme použít flag `-f` abychom se ve výpisu dostali přes revizi s přejmenovaným souborem. Bez flagu se provádění příkazu `log` zastaví u změny názvu:

```
$ hg log y2                        # pouze ty revize, kde je y2, resp. y1
changeset:   6:47445946964f
tag:         tip
user:        Thorbjorn Jemander
date:        Sat Dec 12 17:08:21 2009 +0100
summary:     Updated module y2 with feature C.

changeset:   5:538cc1eb311d
user:        Thorbjorn Jemander
date:        Sat Dec 12 17:05:38 2009 +0100
summary:     y1 renamed to y2

$ hg log -f y2                     # v jiných revizích se y2, y1 nevyskytuje
changeset:   6:47445946964f
tag:         tip
user:        Thorbjorn Jemander
date:        Sat Dec 12 17:08:21 2009 +0100
summary:     Updated module y2 with feature C.

changeset:   5:538cc1eb311d
user:        Thorbjorn Jemander
date:        Sat Dec 12 17:05:38 2009 +0100
summary:     y1 renamed to y2

changeset:   1:01e9f6a4aae5
user:        Thorbjorn Jemander
date:        Sat Dec 12 10:12:09 2009 +0100
```

```
summary:     Added A to x2 and B to y1.
```

Rozdíl mezi verzemi souboru zjistíme příkazem `diff`:

```
$ echo D > y2

$ hg diff y2                    # Rozdíl mezi tipem a pracovní kopií
diff -r 47445946964f y2
--- a/y2  Sat Dec 12 17:08:21 2009 +0100
+++ b/y2  Sat Dec 12 17:22:41 2009 +0100
@@ -1,1 +1,1 @@
-C
+D

$ hg diff -r 5 y2               # Rozdíl mezi revizí 5 a pracovní kopií
diff -r 538cc1eb311d y2
--- a/y2  Sat Dec 12 17:05:38 2009 +0100
+++ b/y2  Sat Dec 12 17:23:41 2009 +0100
@@ -1,1 +1,1 @@
-B
+D

$ hg cat -r 5 y2                # Vytisknout y2 s obsahem v revizi 5.
B
$ hg ci -m"Let advance to experimental D version"
```

Poslední příklad s `hg cat` lze použít k výpisu souboru z historie a přesměrování jeho obsahu do lokálního (nalézajícího se v aktuální pracovní kopii) souboru:

```
$ hg tip
changeset:   7:5090879191a2
tag:         tip
user:        Thorbjorn Jemander
date:        Sat Dec 12 17:31:47 2009 +0100
summary:     Let advance to experimental D version
$ hg cat -r 6 y2 > y2            # Přenést obsah y2 v revizi 6
                                 # do souboru y2 v revizi tip
$ hg diff
diff -r 5090879191a2 y2
--- a/y2  Sat Dec 12 17:31:47 2009 +0100
+++ b/y2  Sat Dec 12 17:34:01 2009 +0100
@@ -1,1 +1,1 @@
-D
+C
$ hg ci -m"D caused a thermonuclear explosion. Backed to C."
```

Takto lze poněkud primitivním způsobem vrátit předchozí změnu.. Lepší způsoby si vysvětlíme v dalším odstavci.

4.3 Zpět: revert, forget a rollback

Poslední příklad ukazoval způsob návratu k předchozímu stavu pomocí příkazu `cat`. Existuje však

příkaz – `revert` – který je pro tento účel přímo určený (a pracuje rovněž s adresáři):

```
$ hg revert -r5 y2        # Navrátí lokální kopii y2 do stavu revize 5
$ cat y2
B
$ hg ci -m"y2/C caused a chemical explosion. Going back to y2/B."
```

Co se však stane, když náhodou přidáme soubory a chceme je ještě před zápisem do repozitáře odebrat? Jednoduše řekneme Mercurialu, aby na ně zapomněl:

```
$ touch a y2.o z1.o  #  Vytvoří se 3 soubory

$ hg addremove
adding a
adding y2.o
adding z1.o
$ hg forget y2.o z1.o
$ hg ci -m"Adding new module a"
$ rm *.o
```

Chcete vždycky ignorovat soubory ~.o? Můžete určit jejich vzory v souboru .hgignore v pracovním adresáři. Více v manové stránce pro hgignore: man hgignore.

Předpokládejme, že jsme komitovali něco velmi hloupého, čeho litujeme. Naštěstí je možné z poslední transakce vycouvat (rollback). Nahlížejme na `rollback` jako na zrušení posledního komitu:

```
$ mv * /tmp
...

$ hg addremove                    # Možná jste nesoustředěni...
removing a
removing y2
removing z1
$ hg ci -m"quick check-in"        # ... a předáte repozitáři něco špatného.
$ ls                              # ... všechny změny byly zapsány
$ hg rollback                     # Nevadí, vykulíme je zpět!
rolling back last transaction
$ hg up -C                        # Potřebujeme aktualizovat pracovní kopii.
3 files updated, 0 files merged, 0 files removed, 0 files unresolved
$ ls
a y2 z1                           # Hle a máme to zpátky.
```

Pamatujme si, že po hg rollback potřebujeme aktualizovat pracovní kopii. Věžme, že jsme se nejenom vrátili v historii o jeden krok, ale že jsme poslední záznam z historie úplně vymazali.

Příkaz rollback se nedá opakovat vícekrát za sebou, protože Mercurial ukládá informace jenom pro jeden rollback. V kapitole 9.2 pohovoříme o ještě účinnějších technikách.

4.4 Hledání: grep a locate

V této chvíli umíme provádět commit, zrušit operaci a pohybovat se v historii revizí dozadu i vpřed. Následujícím tematem bude vyhledání souboru a jeho obsahu v historii. Pro tento účel jsou k disposici dva příkazy: `grep` a `locate`. Pracují podobně jako stejnojmenné příkazy v systému UNIX. U následujícího příkladu si uvědomíme, že soubor *y1* byl již dříve přejmenován na *y2* a že soubor *y2* měnil svůj obsah v průběhu historie (B → C → D → C → B).

```
$ hg locate y1              # Soubor y1 v aktuální revizi není
$ hg locate -r 1 y1         # Je však v revizi 1
y1
$ hg grep B y2              # Která poslední revize souboru y2 obsahuje B?
y2:9:B                      # revize 9
$ hg grep --all B y2        # Kde všude se B objevilo?
y2:9:+:B                    # B se objevilo v 9
y2:6:-:B                    # B zmizelo v 6
y2:5:+:B                    # B se poprvé objevilo v 5
$ hg grep -f --all B y2     # Flag -f nás dostane
y2:9:+:B                    #  přes změnu jména y2.
y2:6:-:B
y2:5:+:B
y1:1:+:B
```

Chcete vědět, kdo je zodpovědný za určitý řádek v souboru? Můžete to zjistit použitím příkazu "blame", nebo "annotate". Více viz "hg help annotate".

Flag `--all` znamená tiskni všechny shody, zatímco `-f` znamená *follow renames* (jdi za změnu jména).

4.4.1 Bisect

Předpokládejme, že jsme dostali od zákazníka zprávu o chybě a chtěli bychom vědět kdy tato chyba vznikla, abychom si mohli udělat představu o počtu postižených vydání. Mercurial nám pomůže nalézt počátek chyby, pokud mu poskytneme *testovací příkaz*. Testovací příkaz vrací 0, není-li chyba nalezena a 1, je-li chyba zjištěna. Tento testovací příkaz používá Mercurial skákaje v historii vpřed a vzad při hledání *dobrých* či *špatných* revizí. Tuto činnost ukončí, jakmile určí nejstarší špatnou revizi.

Představme si například, že máme chybu ve svém systému, která spočívá v tom, že soubory *y* a *z* mají rozdílná čísla: současná existence souborů y2 a z1 představuje chybový stav, zatímco *y1* a *z1* nikoliv. Testovací funkce, implementovaná jako *shell script*, může vypadat podobně jako zápis v rámečku.

```
#!/bin/sh
if [ z* = z1 -a y* = y1 ]; then
   echo No bug!
   exit 0
else
   echo Bug!
   exit 1
fi
```

The file test.sh

Překopírujme tento skript do souboru **test.sh** a pomocí chmod jej učiňme spustitelným. Spusťme jej a uvidíme, že chyba se nalézá ve stávající pracovní kopii.

Aby mohl Mercurial nalézt chybu (bug), potřebuje od nás tři informace:

1. Číslo revize obsahující chybu (`hg bisect -b`),
2. Číslo revize neobsahující chybu (`hg bisect -g`),
3. Testovací příkaz (`hg bisect -c`)

Víme jistě, že changeset 3 chybu neobsahuje a stávající revize (10) ano.

Zadejme následující příkazy a Mercurial spustí testovací skript u zadaného výběru revizí a určí, ve které revizi se chyba vyskytuje poprvé:

```
$ hg bisect -r      # Započetí procedury bisect
$ hg bisect -b 10   # Revize 10 (tip) obsahuje bug
$ hg bisect -g 3    # Revize 3 nikoliv.
Testing changeset 6:1f6db2bf99f3 (7 changesets remaining, ~2 tests)
1 files updated, 0 files merged, 1 files removed, 0 files unresolved
$ hg bisect -c ./test.sh
Bug!
Changeset 6:47445946964f: bad
No bug!
Changeset 4:4207f3b7f4ee: good
Bug!
Changeset 5:538cc1eb311d: bad
The first bad revision is:           # První špatná revize je:
changeset:   5:538cc1eb311d
user:        Thorbjorn Jemander
date:        Sat Dec 12 17:05:38 2009 +0100
summary:     y1 renamed to y2
$ rm test.sh
```

Chyba byla zavedena v revizi 5, kde jsme přejmenovali *y1* na *y2*. V tomto případě jsme první výskyt chyby mohli nalézt prohlédnutím logu ale obvykle to nebývá tak snadné.

5 Dělení historie

Než postoupíme k dalšímu tematu, přidáme k Mercurialu extenzi **graphlog**, která nám umožní snadnější prohlížení historie. V našem domovském adresáři otevřeme (nebo vytvoříme) soubor *hgrc*.

```
[ui]
username = Thorbjorn Jemander

[extensions]
graphlog=
```

The file ~/.hgrc

Vpravo vidíme zadané jméno uživatele a povolenou extenzi graphlog. Soubor uložíme a vyzkoušíme příkazem `glog` :

```
$ hg glog
o  changeset:   10:07c3c9ade9e5
|  tag:         tip
|  user:        Thorbjorn Jemander
|  date:        Sun Dec 13 13:15:37 2009 +0100
|  summary:     Adding new module a
|
o  changeset:   9:1c03fc41fe77
|  user:        Thorbjorn Jemander
|  date:        Sat Dec 12 17:46:52 2009 +0100
|  summary:     y2/C caused a chemical explosion. Going back to y2/B.
|
o  changeset:   8:6ff8f0a0aa98
|  user:        Thorbjorn Jemander
|  date:        Sat Dec 12 17:35:29 2009 +0100
|  summary:     D caused a thermonuclear explosion. Backed to C.
|
o  changeset:   7:5090879191a2
|  user:        Thorbjorn Jemander
|  date:        Sat Dec 12 17:31:47 2009 +0100
|  summary:     Let advance to experimental D version
...
```

Vidíme-li vlevo svislou čárkovanou čáru, naše instalace extenze graphlog byla úspěšná. Podobně můžeme povolit řadu dalších extenzí, ale o tom si povíme postupně později.

Uvažme následující scénář. Před nějakým časem (v revizi 3) jsme zákazníkovi uvolnili verzi R1.0 a nyní pracujeme na verzi R2.0, přičemž jsme provedli řadu změn. Náhle nám zákazník oznámí chybu ve verzi R1.0 a vyžaduje rychlou nápravu. Verze R2.0 není pro vydání ještě zralá, musíme se tedy vrátit a opravit verzi R1.0.

Pohleďme, co se při návratu k verzi R1.0 (changeset 3) stane:

```
$ hg update -r 3    # Běž k revizi 3
1 files updated, 0 files merged, 2 files removed, 0 files unresolved
$ ls                # Vypiš obsah pracovní kopie
y1 z1
$ cat y1            # Vypiš obsah y1
B                   # Ha! Mělo to být C!
$ echo C > y1       # Takže jsme chybu opravili
$ hg ci -m"Fixed the R1.0 bug."    # a předali repozitáři
created new head                   # přičemž jsme vytvořili nové čelo (head)!
```

Cože to "nové čelo" znamená? Budeme-li citovat film “Zpátky k budoucnosti", vytvořili jsme právě “alternativní budoucnost" neboli “alternativní časovou linii". Místo jedné přímé linie od první revize k poslední máme nyní od revize 3 linie dvě. Zobrazíme si to příkazem graphlog:

```
$ hg glog
@  changeset:   11:47da354e2d4d
|  tag:         tip
|  parent:      3:c9d133e1c054
|  user:        Thorbjorn Jemander
|  date:        Tue Dec 15 22:14:59 2009 +0100
|  summary:     Fixed the R1.0 bug.
|
| o  changeset:   10:07c3c9ade9e5
| |  user:        Thorbjorn Jemander
| |  date:        Sun Dec 13 13:15:37 2009 +0100
| |  summary:     Adding new module a
| |
| o  changeset:   9:1c03fc41fe77
| |  user:        Thorbjorn Jemander
| |  date:        Sat Dec 12 17:46:52 2009 +0100
| |  summary:     y2/C caused a chemical explosion. Going back to y2/B.
| |
...
| o  changeset:   5:538cc1eb311d
| |  user:        Thorbjorn Jemander
| |  date:        Sat Dec 12 17:05:38 2009 +0100
| |  summary:     y1 renamed to y2
| |
| o  changeset:   4:4207f3b7f4ee
|/   user:        Thorbjorn Jemander
|    date:        Sat Dec 12 16:41:03 2009 +0100
|    summary:     Added tag my-x1-removal for changeset 3d8afaf1ac30
|
o  changeset:   3:c9d133e1c054
|  user:        Thorbjorn Jemander
|  date:        Sat Dec 12 10:15:49 2009 +0100
|  summary:     Adding C to z1 and removing x2
...
```

Máme nyní dvě *čela,* changeset 10 a 11. *Rodičem* changesetu 10 je 9, zatímco rodičem revize 11 je 3. Změnou pracovní kopie starší revize se implicitně vytváří další (bezejmenná) větev. *Společným předkem* obou větví je v našem případě changeset 3..

5.1 Čela

Výpis čel v našem repozitáři nám zprostředkuje příkaz `heads`:

```
$ hg heads
changeset:   11:47da354e2d4d
tag:         tip
parent:      3:c9d133e1c054
user:        Thorbjorn Jemander
date:        Tue Dec 15 22:14:59 2009 +0100
summary:     Fixed the R1.0 bug.
```

```
changeset:   10:07c3c9ade9e5
user:        Thorbjorn Jemander
date:        Sun Dec 13 13:15:37 2009 +0100
summary:     Adding new module a
```

5.2 Sloučení

Můžeme mít jedno, dvě nebo více čel. Z praktických důvodů bychom se však měli snažit mít vždycky jen jedno čelo. Dvě větve sloučíme příkazem `merge`:

```
$ hg merge
merging y1 and y2 to y2
2 files updated, 1 files merged, 0 files removed, 0 files unresolved
(branch merge, don't forget to commit)
$ hg ci -m"Merged R1.0-fix with pre-2.0"
$ hg glog
@    changeset:   12:90cf941b9def
|\   tag:         tip
| |  parent:      11:47da354e2d4d
| |  parent:      10:07c3c9ade9e5
| |  user:        Thorbjorn Jemander
| |  date:        Tue Dec 15 22:31:43 2009 +0100
| |  summary:     Merged R1.0-fix with pre-2.0
| |
| o  changeset:   11:47da354e2d4d
| |  parent:      3:c9d133e1c054
| |  user:        Thorbjorn Jemander
| |  date:        Tue Dec 15 22:14:59 2009 +0100
| |  summary:     Fixed the R1.0 bug.
| |
o |  changeset:   10:07c3c9ade9e5
| |  user:        Thorbjorn Jemander
| |  date:        Sun Dec 13 13:15:37 2009 +0100
| |  summary:     Adding new module a
```

Nyní máme opět jen jedno čelo. Changeset 12 má dva rodiče, 11 and 10.

5.3 Větvení

Vhodný způsob pro vytvoření nové větve je s použitím příkazu `branch`, přičemž lze zadat i její symbolické označení. Následující sekvence vytvoří větev R1.5, vycházející z revize 7:

```
$ hg up 7                       # Běž k revizi 7...
1 files updated, 0 files merged, 1 files removed, 0 files unresolved
$ hg branch R1.5                # a vytvoř větev R1.5
marked working directory as branch R1.5
$ hg ci -m"Created R1.5 branch"
created new head
$ hg branches
R1.5                            13:a977411494e5
default                         12:90cf941b9def
$ hg glog
@  changeset:   13:a977411494e5
|  branch:      R1.5
|  tag:         tip
```

Pamatujte si, že tyto větve nejsou lokální, t.zn. že budou exportovány do jiných repozitářů při přenosu změn mezi repozitáři. Nepoužívejte je tedy pro privátní vývoj..

```
|  parent:       7:5090879191a2
|  user:         Thorbjorn Jemander
|  date:         Tue Dec 15 22:39:21 2009 +0100
|  summary:      Created R1.5 branch
|
| o    changeset:   12:90cf941b9def
| |\   parent:      11:47da354e2d4d
| | |  parent:      10:07c3c9ade9e5
| | |  user:        Thorbjorn Jemander
| | |  date:        Tue Dec 15 22:31:43 2009 +0100
| | |  summary:     Merged R1.0-fix with pre-2.0
| | |
....
| | |
| o |  changeset:   9:1c03fc41fe77
| | |  user:        Thorbjorn Jemander
| | |  date:        Sat Dec 12 17:46:52 2009 +0100
| | |  summary:     y2/C caused a chemical explosion. Going back to y2/B.
| | |
| o |  changeset:   8:6ff8f0a0aa98
|/ /   user:        Thorbjorn Jemander
| |    date:        Sat Dec 12 17:35:29 2009 +0100
| |    summary:     D caused a thermonuclear explosion. Backed to C.
| |
o |  changeset:   7:5090879191a2
| |  user:        Thorbjorn Jemander
| |  date:        Sat Dec 12 17:31:47 2009 +0100
| |  summary:     Let advance to experimental D version
```

Nyní opět máme dvě *čela*: 13 a 12. Můžeme přeskakovat od jedné větve ke druhé zadáním jejich názvu jako argumentu pro příkaz `update`:

```
$ hg up default
2 files updated, 0 files merged, 0 files removed, 0 files unresolved
$ hg up R1.5
1 files updated, 0 files merged, 1 files removed, 0 files unresolved

$ hg branch                          # Bez argumentu dostaneme název větve
R1.5
```

Než budeme pokračovat, uzavřeme větev R1.5 a sloučíme ji s hlavní větví, abychom neměli dvě čela:

```
$ hg ci --close-branch -m"Branch R1.5 closed."
2 files updated, 0 files merged, 0 files removed, 0 files unresolved
$ hg up default
2 files updated, 0 files merged, 0 files removed, 0 files unresolved
$ hg merge R1.5
0 files updated, 0 files merged, 0 files removed, 0 files unresolved
(branch merge, don't forget to commit)
$ hg ci -m"R1.5 merged into default."
$ hg branches
default                      15:25ce584f89d6
$ hg glog
@    changeset:   15:25ce584f89d6
|\   tag:         tip
| |  parent:      12:4b9ce0577295
| |  parent:      14:c0ba01e32db9
| |  user:        Thorbjorn Jemander
| |  date:        Sat Dec 19 11:05:54 2009 +0100
| |  summary:     R1.5 merged into default.
| |
| o  changeset:   14:c0ba01e32db9
| |  branch:      R1.5
| |  user:        Thorbjorn Jemander
| |  date:        Sat Dec 19 11:04:35 2009 +0100
| |  summary:     Branch R1.5 closed.
| |
| o  changeset:   13:cdfb8c39e915
| |  branch:      R1.5
| |  parent:      7:bf7c4059b77a
| |  user:        Thorbjorn Jemander
| |  date:        Sat Dec 19 10:51:06 2009 +0100
| |  summary:     Created R1.5 branch
...
```

Někdy nechcete připojit celou větev, pouze určité vybrané změny. Tomu se říká "vybírání třešniček" a lze to provést příkazem "transplant". Viz kapitola 9 Další extenze.

Vězme, že jedinou věc, kterou flag `--close-branch` dělá, je to, že skryje název větve a učiní ji neaktivní. Pokud známe její název, můžeme jej používat v příkazech, protože zůstává platným identifikátorem. Větev bude také viditelná v historii změn, ale nevypíše se po zadání příkazu `hg branches`.

6 Decentralizovaný vývoj

Pokročme k decentralizovanému (distributed) vývoji projektu, to jest k práci s několika repozitáři.

6.1 Klonování

Přesnou repliku repozitáře lze vytvořit příkazem `clone`:

```
$ cd ..
$ hg clone hg1 hg2
updating working directory
4 files updated, 0 files merged, 0 files removed,
 0 files unresolved
$ cd hg2
$ hg glog
@    changeset:   15:25ce584f89d6
|\   tag:         tip
| |  parent:      12:4b9ce0577295
| |  parent:      14:c0ba01e32db9
| |  user:        Thorbjorn Jemander
| |  date:        Sat Dec 19 11:05:54 2009 +0100
| |  summary:     R1.5 merged into default.
| |
| o  changeset:   14:c0ba01e32db9
| |  branch:      R1.5
| |  user:        Thorbjorn Jemander
| |  date:        Sat Dec 19 11:04:35 2009 +0100
| |  summary:     Branch R1.5 closed.
| |
| o  changeset:   13:cdfb8c39e915
| |  branch:      R1.5
| |  parent:      7:bf7c4059b77a
| |  user:        Thorbjorn Jemander
| |  date:        Sat Dec 19 10:51:06 2009 +0100
| |  summary:     Created R1.5 branch
| |
o |    changeset:   12:4b9ce0577295
|\ \   parent:      11:a53cffb1a107
| | |  parent:      10:185eb7291e1d
| | |  user:        Thorbjorn Jemander
| | |  date:        Sat Dec 19 10:50:19 2009 +0100
| | |  summary:     Merged R1.0-fix with pre-2.0
| | |
| o |  changeset:   11:a53cffb1a107
| | |  parent:      3:9277041236ca
...
```

Zde klonujeme lokální kopii repozitáře, ale můžete klonovat i vzdálený repozitář zadáním URL, např. http://, https:// nebo ssh://

Klon má tutéž historii jako originál, včetně větví a tagů. Od originálu se klon liší pouze v jedné věci: klon ví, z kterého repozitáře byl vytvořen; má *implicitní repozitář* (default repository):

```
$ cat .hg/hgrc
.hg/hgrc:default = /home/<user>/hg-tutorial/hg1
```

Složka `.hg` obsahuje všechny "účetnické" informace o repozitáři a soubor hgrc poukazuje (mezi jinými věcmi) na implicitní repozitář. Je možné provádět příkazy *pull* a *push* mezi repozitáři a pokud není zadán explicitní argument, je předpokládán implicitní repozitář:

```
$ hg incoming                       # prozkoumá implicitní repozitář.
comparing with /home/thorman/hg-tutorial/hg1
searching for changes
no changes found
$ cd ../hg1
$ hg incoming                       # hg1 nemá implicitní repozitář.
abort: repository default not found!
```

6.2 Pull

Začneme přesunovat změny mezi dvěma repozitáři:

```
$ cd ../hg1
$ echo "new feature" > b

$ hg addremove
adding b
$ hg ci -m"added new feature to b"
$ hg glog
@  changeset:   16:cfa13befd88e
|  tag:         tip
|  user:        Thorbjorn Jemander
|  date:        Sat Dec 19 11:16:17 2009 +0100
|  summary:     added new feature to b
|
o    changeset:   15:25ce584f89d6
|\   parent:      12:4b9ce0577295
| |  parent:      14:c0ba01e32db9
| |  user:        Thorbjorn Jemander
| |  date:        Sat Dec 19 11:05:54 2009 +0100
| |  summary:     R1.5 merged into default.
| |
| o  changeset:   14:c0ba01e32db9
| |  branch:      R1.5
| |  user:        Thorbjorn Jemander
| |  date:        Sat Dec 19 11:04:35 2009 +0100
| |  summary:     Branch R1.5 closed.
...
$ cd ../hg2
$ hg incoming                 # Ověřme si, zda v hg1 nejsou nějaké změny ..
comparing with /home/thorman/hg-tutorial/hg1
searching for changes
changeset:   16:cfa13befd88e
tag:         tip
user:        Thorbjorn Jemander
date:        Sat Dec 19 11:16:17 2009 +0100
summary:     added new feature to b
$ hg pull                     # Zjistili jsme, že ano - tak si je stáhneme.
pulling from /home/thorman/hg-tutorial/hg1
searching for changes
adding changesets
adding manifests
adding file changes
added 1 changesets with 1 changes to 1 files
(run 'hg update' to get a working copy)
$ ls
a  y2  z1                     # Jakpak to? Žádný soubor b?
$ hg up                       # Aha, nutno provést update.
```

```
1 files updated, 0 files merged, 0 files removed, 0 files unresolved
$ ls
a  b  y2  z1
```

6.2.1 Překážející nekomitované změny

Někdy se nalézáme uprostřed velké předělávky a z nějakých důvodů nechceme nic komitovat, ale přesto chceme stahovat změny z jiných repozitářů. Obvykle se nám daří stahovat změny, slučovat je a aktualizovat pracovní kopii, ale někdy se setkáme s následujicím chybovým hlášením:

`abort: outstanding uncommitted changes`

Tato diagnostika znamená, že má Mercurial problémy při slučování nekomitovaných změn. To může čas od času způsobit velké bolení hlavy. Jsou tři způsoby řešení (o nichž vím) tohoto problému:

1. Vždy provádět commit před stahováním změn. To se však lehčeji řekne než udělá. Někdy se nehodí stávající změny komitovat. Důvody mohou být různé.
2. Pro importování změn používat oddělený repozitář, tak, jak je popsáno zde: http://blogs.sun.com/tor/entry/mercurial_tip_checking_in_regularly . To funguje, ale je to poněkud nepraktické.
3. Třetí způsob (jež preferuji) je dočasně odklidit lokální změny, provést pull a merge externích changesetů a posléze znovu použít lokální změny.

Jak a proč k tomu dochází je podrobně popsáno v kapitole 8. Jsou tam rovněž uvedeny praktické způsoby řešení tohoto problému.

Následuje opak k pull: push.

6.3 Push

Změny můžeme poslat zpět do původního repozitáře příkazem `push:`

```
$ echo "New module: c" > c            # Jsme v adresáři hg2
$ hg add c
$ hg ci -m"Added the new module c"
$ hg outgoing                          # Odchozí změny ...
comparing with /home/thorman/hg-tutorial/hg1
searching for changes
changeset:   17:8e48e69496ec
tag:         tip
user:        Thorbjorn Jemander
date:        Sat Dec 19 11:19:02 2009 +0100
summary:     Added the new module c
$ hg push                        # .. posíláme do implicitního repozitáře
pushing to /home/thorman/hg-tutorial/hg1
searching for changes
adding changesets
adding manifests
adding file changes
added 1 changesets with 1 changes to 1 files
$ cd ../hg1
$ ls
a b y2 z1                      # Cože? Žádný modul c?
```

```
$ hg up                          # Ach, opět zapomněli na update.
1 files updated, 0 files merged, 0 files removed, 0 files unresolved
$ ls
a b c y2 z1
```

Když provedeme push, nevidí pracovní kopie přijemce přijmuté změny automaticky, nýbrž až po její aktualizaci (což je totožné s chováním pracovní kopie při pull).

To byl jednoduchý případ. Nyní předpokladejme, že dva vývojáři pracují současně na pracovních adresářích hg1 a hg2 a chtějí si vyměňovat své kódy:

```
$ echo "A new nifty feature" > b      # Jsme v adresáři hg1
$ hg ci -m"New feature added to b."
$ cd ../hg2
$ echo "Bugfix" > c
$ hg ci -m"Fixed a bug in c."
$ hg incoming
comparing with /home/thorman/hg-tutorial/hg1
searching for changes
changeset:   18:ef562bd14249
tag:         tip
user:        Thorbjorn Jemander
date:        Sat Dec 19 11:24:18 2009 +0100
summary:     New feature added to b.
$ hg pull
pulling from /home/thorman/hg-tutorial/hg1
searching for changes
adding changesets
adding manifests
adding file changes
added 1 changesets with 1 changes to 1 files (+1 heads)
(run 'hg heads' to see heads, 'hg merge' to merge)
```

Hmm... pročpak nám Mercurial říká "`+1 heads`" (čela)? Důvodem je to, že změny v hg1 i v hg2 byly založeny na changesetu 17, takže když jsme stáhli změny z hg1, měli jsme náhle dvě čela, 18 a 19:

```
$ hg glog
o  changeset:   19:ef562bd14249
|  tag:         tip
|  parent:      17:8e48e69496ec
|  user:        Thorbjorn Jemander
|  date:        Sat Dec 19 11:24:18 2009 +0100
|  summary:     New feature added to b.
|
| @  changeset:   18:a7ba278d6de2
|/   user:        Thorbjorn Jemander
|    date:        Sat Dec 19 11:24:53 2009 +0100
|    summary:     Fixed a bug in c.
|
o  changeset:   17:8e48e69496ec
|  user:        Thorbjorn Jemander
|  date:        Sat Dec 19 11:19:02 2009 +0100
|  summary:     Added the new module c
...
$ hg heads
changeset:   19:ef562bd14249
```

```
tag:         tip
parent:      17:8e48e69496ec
user:        Thorbjorn Jemander
date:        Sat Dec 19 11:24:18 2009 +0100
summary:     New feature added to b.

changeset:   18:a7ba278d6de2
user:        Thorbjorn Jemander
date:        Sat Dec 19 11:24:53 2009 +0100
summary:     Fixed a bug in c.
```

Potvrdil nám to i příkaz `hg heads`.

Tato čela potřebujeme sloučit a teprve potom můžeme poslat změny do hg1:

```
$ hg merge
1 files updated, 0 files merged, 0 files removed, 0 files unresolved
(branch merge, don't forget to commit)
$ hg ci -m"Merged in hg1 changes."
$ hg push
pushing to /home/thorman/hg-tutorial/hg1
searching for changes
adding changesets
adding manifests
adding file changes
added 2 changesets with 1 changes to 1 files
$ cd ../hg1
$ hg up
1 files updated, 0 files merged, 0 files removed, 0 files unresolved
```

Sloučení potřebujeme provést pokaždé, když stáhneme změny z jiného repozitáře a ve svém repo máme (komitované) změny. To je nezbytné proto, že v obou repozitářích jsou nezávislé linky vývoje. Jiné SCM aplikace tento fakt neodhalují, ale Mercurial ano. Znamená to, že musíme provádět ručně příkaz merge pokaždé, když provedeme pull? Nikoliv, jak uvidíme později, lze to automatizovat. Nejprve se ale zaměřme na několik jiných témat.

6.3.1 Omezení

Jsme nyní v *hg2*, připojili jsme sloučením nový prvek *b* a předali jsme uživateli *hg1* opravu chyby v *c*. Uživatel *hg1* mezitím zapracoval a přidal další prvek do *b*:

```
$ cd ../hg1
$ echo "another nice feature" >> b        # Nepřehlédněte zdvojené >>!
$ hg ci -m"Added another feature to b."
```

A uživatel *hg2* byl také pilný:

```
$ cd ../hg2
$ echo "another bugfix in c" >> c         # .. nenahradí, ale přidá
$ hg ci -m"Another bugfix to c."
```

Uživatel *hg2* se nyní snaží sdílet svou opravu chyby s uživatelem *hg1* posláním změny:

```
$ hg push
pushing to /home/thorman/hg-tutorial/hg1
searching for changes
abort: push creates new remote heads!
(did you forget to merge? use push -f to force)
```

Mercurial odmítl provést zadaný příkaz, protože by push vytvořil ve vzdáleném repozitáři nové čelo. Proč? Prohlédněme si oba záznamy:

```
$ cd ../hg1
$ hg glog
@  changeset:   21:7c33c9969d45
|  tag:         tip
|  user:        Thorbjorn Jemander
|  date:        Mon Dec 28 14:00:43 2009 +0100
|  summary:     Added another feature to b.
|
o    changeset:   20:68ee78138e47
|\   parent:      19:bf134b53f0f0
| |  parent:      18:c22bd5b3858a
| |  user:        Thorbjorn Jemander
| |  date:        Mon Dec 28 13:44:23 2009 +0100
| |  summary:     Merged in hg1 changes.

...
```

Zde vidíme, že changeset 20 (“Merged in hg1 changes") má dítě 21 (“Added another feature to b.").

Na záznamu z hg2 vidíme,

```
$ cd ../hg2
$ hg glog
@  changeset:   21:d8ab303e592a
|  tag:         tip
|  user:        Thorbjorn Jemander
|  date:        Mon Dec 28 14:02:13 2009 +0100
|  summary:     Another bugfix to c.
|
o    changeset:   20:68ee78138e47
|\   parent:      18:bf134b53f0f0
| |  parent:      19:c22bd5b3858a
| |  user:        Thorbjorn Jemander
| |  date:        Mon Dec 28 13:44:23 2009 +0100
| |  summary:     Merged in hg1 changes.
...
```

že *hg2* má tentýž changeset 20 (“Merged in hg1 changes") s dítětem 21, ale je to odlišný changeset (“Another bugfix to c."). Kdybychom naléhali na provedení *push*, měl by changeset 20 dvě děti a tím bychom dostali dvě čela.

Vytváření dalších čel ve vzdálených repozitářích je v Mercurialu považováno za nezdvořilé (nebo nepraktické); čelo je dovoleno vytvořit jenom při stahování změn:

```
$ cd ../hg1
$ hg pull ../hg2
```

```
pulling from ../hg2
searching for changes
adding changesets
adding manifests
adding file changes
added 2 changesets with 1 changes to 2 files (+1 heads)
(run 'hg heads' to see heads, 'hg merge' to merge)
$ hg merge
1 files updated, 0 files merged, 0 files removed, 0 files unresolved
(branch merge, don't forget to commit)
$ hg ci -m"Merged in hg2 changes."
```

Alternativně stáhneme změny *hg1* do *hg2*, sloučíme je a výsledek potom strčíme zpátky do *hg1*, ale nyní to zatím nedělejme, protože by nám to pomíchalo situaci pro další příklad.

6.4 Slučování konfliktů

Nejprve nainstalujeme nástroj pro slučování (pokud není již součástí naší aplikace). Použiji zde kdiff3.

```
$ sudo apt-get install kdiff3
```

Potom upravíme soubor ~/.hgrc tak, aby obsahoval nové položky podle obrázku dole (říkáme Mercurialu, aby použil kdiff3 jako slučovací nástroj).

```
[ui]
username = Thorbjorn Jemander

[extensions]
graphlog=
hgext.extdiff =

[extdiff]
cmd.kdiff3 =

[merge-tools]
kdiff3.args = $base $local $other -o $output
```

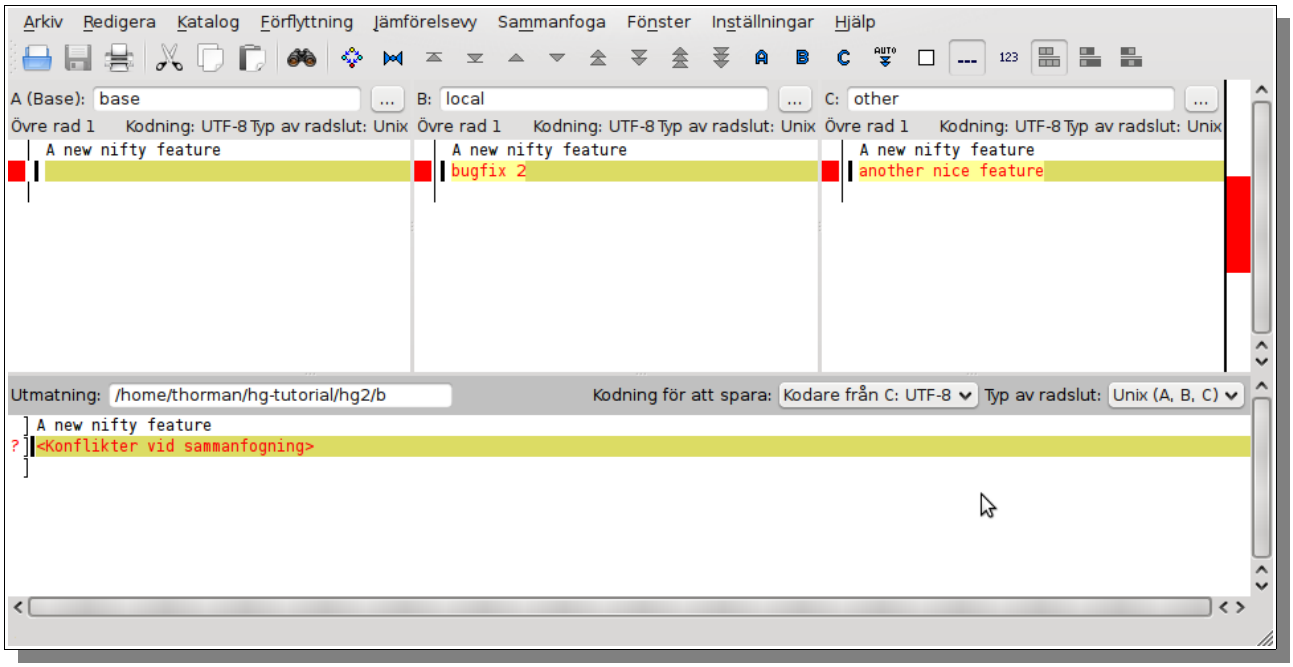
The file ~/.hgrc

Vyzkoušejme kdiff3 na konfliktu, který vytvoříme mezi *hg1* a *hg2* změnou souboru. *Hg2* přidá "bugfix no2" do modulu *b,* což bude v konfliktu s changesetem "another feature" v *hg1*.

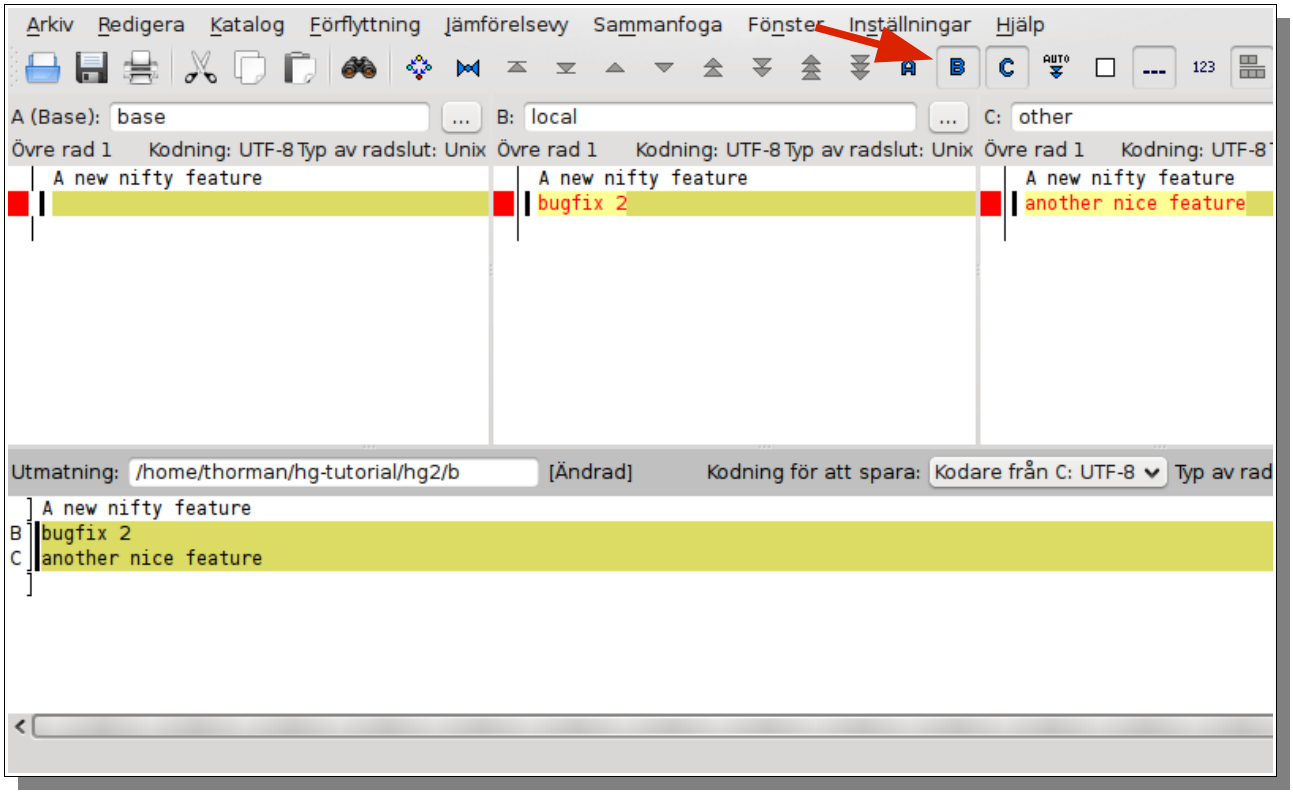
```
$ cd ../hg2
$ echo "bugfix 2" >> b
$ hg ci -m"Bug no 2 fixed in b."
$ hg pull
...
$ hg merge
```

```
merging b
```

Nyní Mercurial vidí, že jak *hg1*, tak *hg2* změnili soubor *b* způsobem, který nejde automaticky sloučit. Proto vyvolá nástroj kdiff3 pro automatické sloučení:



Otevře se okno se čtyřmi panely; třemi v horní části, jedním ve spodní. Panel vlevo (A) ukazuje obsah *báze*, to jest posledního společného předka. Panel B ukazuje lokální změny a panel C ukazuje změny vzdálené verze, kterou chceme připojit. Konflikty jsou označeny červeně. V B vidíme "bugfix 2", zatímco v C "another nice feature". Spodní panel ukazuje výsledek sloučení. Řekněme, že chceme akceptovat obě změny. Poklepeme na tlačítko B a C v nástrojové liště:



Máme nyní vše, co jsme si přáli.

Uložíme, zavřeme a provedeme commit:

```
...
merging b
1 files updated, 1 files merged, 0 files removed, 0 files unresolved
(branch merge, don't forget to commit)
$ hg ci -m"Merged bugfix 2 and another nice feature."
```

V reálném životě jsou konflikty mnohem složitější, ale princip je stejný.

6.5 Fetch

Musíme vždycky provádět postupně `hg pull`, potom `hg merge a nakonec hg commit`? Nikoliv. existuje extenze `fetch`, která to udělá za nás. Upravme soubor ~/.hgrc přidáním řádku "`fetch=`", abychom tuto šikovnou extenzi povolili:

```
[ui]
username = Thorbjorn Jemander

[extensions]
graphlog=
hgext.extdiff =
fetch=

[extdiff]
cmd.kdiff3 =

[merge-tools]
kdiff3.args = $base $local $other -o $output
```

The file ~/.hgrc

Nyní to vyzkoušíme:

```
$ cd ../hg1
$ echo "cleanup" > a
$ hg ci -m"Cleaned up module a."
$ hg incoming
comparing with ../hg2
searching for changes
changeset:   22:917ec356d233
user:        Thorbjorn Jemander
date:        Mon Dec 28 14:18:23 2009 +0100
summary:     Bug no 2 fixed in b.

changeset:   25:d7c7adab510e
tag:         tip
parent:      22:917ec356d233
parent:      24:58f714a8f24c
user:        Thorbjorn Jemander
date:        Mon Dec 28 14:22:15 2009 +0100
summary:     Merged bugfix 2 and another nice feature.
$ hg fetch ../hg2
pulling from ../hg2
searching for changes
adding changesets
adding manifests
adding file changes
added 2 changesets with 2 changes to 1 files (+1 heads)
updating to 26:d7c7adab510e
2 files updated, 0 files merged, 0 files removed, 0 files unresolved
merging with 24:a836f48b2edb
2 files updated, 0 files merged, 0 files removed, 0 files unresolved
new changeset 27:eb831a2a7cef merges remote changes with local
$ cat b
A new nifty feature
bugfix 2
another nice feature
$ hg glog
@    changeset:   27:eb831a2a7cef
|\   tag:         tip
```

```
| |  parent:      26:d7c7adab510e
| |  parent:      24:a836f48b2edb
| |  user:        Thorbjorn Jemander
| |  date:        Mon Dec 28 14:29:17 2009 +0100
| |  summary:     Automated merge with file:///home/thorman/hg-tutorial/hg2
| |
| o    changeset:   26:d7c7adab510e
| |\   parent:      25:917ec356d233
| | |  parent:      23:58f714a8f24c
| | |  user:        Thorbjorn Jemander
| | |  date:        Mon Dec 28 14:22:15 2009 +0100
| | |  summary:     Merged bugfix 2 and another nice feature.
| | |
| | o  changeset:   25:917ec356d233
| | |  parent:      22:d8ab303e592a
| | |  user:        Thorbjorn Jemander
| | |  date:        Mon Dec 28 14:18:23 2009 +0100
| | |  summary:     Bug no 2 fixed in b.
...
```

Máte nepřehlednou historii? Můžete ji zpřehlednit *přeskupením*. Viz *rebase* v kapitole 9.

Nahoře vidíme, jak příkaz fetch provádí pull, merge (je-li zapotřebí) a doplňuje (aktualizuje) lokální kopii. Velice šikovné. Nepřehlédněte automatický komentář komitu počínající "Automated merge..".

6.6 Nepříbuzné repozitáře

Mercurial implicitně nedovoluje akci fetch z nepříbuzných repozitářů, ale je možné toto chování vynutit. Vytvořme dva nepříbuzné repozítáře a uvidíme, co se bude dít:

```
$ mkdir hg3 hg4
$ cd hg3
$ hg init
$ echo a  > a; hg add a; hg ci -m"adding a"
$ echo c1 > c; hg add c; hg ci -m"adding c"
$ cd ../hg4
$ hg init
$ echo b  > b; hg add b; hg ci -m"adding b"
$ echo c2 > c; hg add b; hg ci -m"adding c"
$ hg pull ../hg3
pulling from ../hg3
searching for changes
abort: repository is unrelated
$ hg pull -f ../hg3             # použijeme -f jako force, vynutit
pulling from ../hg3
searching for changes
warning: repository is unrelated
adding changesets
adding manifests
adding file changes
added 2 changesets with 2 changes to 2 files (+1 heads)
(run 'hg heads' to see heads, 'hg merge' to merge)
$ hg glog
o  changeset:   3:3e236b6d437f
|  tag:         tip
|  user:        Thorbjorn Jemander
|  date:        Mon Dec 28 15:05:15 2009 +0100
```

```
|   summary:     adding c
|
o   changeset:   2:1f36da6d72b9
    parent:      -1:000000000000
    user:        Thorbjorn Jemander
    date:        Mon Dec 28 15:03:21 2009 +0100
    summary:     adding a

@   changeset:   1:3d6cdd4bfea2
|   user:        Thorbjorn Jemander
|   date:        Mon Dec 28 15:05:52 2009 +0100
|   summary:     adding c
|
o   changeset:   0:6bb7f0241f12
    user:        Thorbjorn Jemander
    date:        Mon Dec 28 15:05:33 2009 +0100
    summary:     adding b
```

Poté, co jsme vynuceným pull přenesli změny, máme nyní dvě čela, každé z jiného repozitáře, ale také dva changesety bez rodičů (0 a 2). Máme dva kořeny (*roots*), což je neortodoxní.

Sloučení:

```
$ hg merge                    # objeví se kdiff  → merge, save a close.
merging c
1 files updated, 1 files merged, 0 files removed, 0 files unresolved
(branch merge, don't forget to commit)
$ hg ci -m"merged"
$ glog
@     changeset:   4:b36b10f6d2c8
|\    tag:         tip
| |   parent:      1:3d6cdd4bfea2
| |   parent:      3:3e236b6d437f
| |   user:        Thorbjorn Jemander
| |   date:        Mon Dec 28 15:07:10 2009 +0100
| |   summary:     merged
| |
| o   changeset:   3:3e236b6d437f
| |   user:        Thorbjorn Jemander
| |   date:        Mon Dec 28 15:05:15 2009 +0100
| |   summary:     adding c
| |
| o   changeset:   2:1f36da6d72b9
|     parent:      -1:000000000000
|     user:        Thorbjorn Jemander
|     date:        Mon Dec 28 15:03:21 2009 +0100
|     summary:     adding a
|
o   changeset:   1:3d6cdd4bfea2
|   user:        Thorbjorn Jemander
|   date:        Mon Dec 28 15:05:52 2009 +0100
|   summary:     adding c
|
o   changeset:   0:6bb7f0241f12
    user:        Thorbjorn Jemander
    date:        Mon Dec 28 15:05:33 2009 +0100
    summary:     adding b
```

Slučování z nepříbuzných repozitářů bychom se měli vyhýbat, protože to může být složité v případě mnoha konfliktů. Oba stromy nemají společného předka a tudíž žádnou bázovou revizi pro srovnání.

7 Import/Export

V tomto odstavci si ukážeme, jak můžeme komunikovat se světem mimo Mercurial.

7.1 Archiv

Tu a tam potřebujeme poslat zdrojový kód někomu, koho historie revizí nezajímá. Vhodným nástrojem je příkaz `archive`:

```
$ cd hg1
$ hg archive -t zip ../hg1.zip
$ cd ..
$ ls -l hg1.zip
-rw-r--r--  1 thorman thorman  890 2009-12-28 15:24 hg1.zip
```

Dostaneme archivní soubor, který obsahuje zdrojový kód pouze aktuální revize. Můžeme určit několik archivovacích typů (“files", “tar", “tgz", “zip" atp).

7.2 Export

Jindy nechceme exportovat celý zdrojový kód, ale jenom jeho část, nazývanou *patch* (oprávka). Oprávka je soubor, popisující změny jednoho či více souborů. Pro vytvoření oprávky z changesetu použijeme příkaz `export`:

```
$ cd hg1
$ echo “testing patch-gen" > p
$ hg addr
adding p
$ hg ci -m"added p"
$ hg head
changeset:   28:0d8301efd4b4
tag:         tip
user:        Thorbjorn Jemander
date:        Mon Dec 28 15:39:30 2009 +0100
summary:     adding p
$ hg export -o ../hg1-rev28.diff 28
$ cat ../hg1-rev28.diff
# HG changeset patch
# User Thorbjorn Jemander
# Date 1262011170 -3600
# Node ID 0d8301efd4b47bc4dbed24353a8f21632fe0876a
# Parent  eb831a2a7cefd421aca032af6ccabc7fbefc4e71
adding p

diff -r eb831a2a7cef -r 0d8301efd4b4 p
--- /dev/null Thu Jan 01 00:00:00 1970 +0000
+++ b/p  Mon Dec 28 15:39:30 2009 +0100
@@ -0,0 +1,1 @@
+testing patch-gen
```

7.3 Import

Oprávky můžeme včlenit do svého repozitáře příkazem `import`:

```
$ cd hg2
$ hg import ../hg1-rev28.diff
applying ../hg1-rev28.diff
$ hg head
changeset:   26:21fdaa10b77d
tag:         tip
user:        Thorbjorn Jemander
date:        Mon Dec 28 15:39:30 2009 +0100
summary:     adding p
$ cat p
testing patch-gen
```

Importováním se oprávka stala v hg2 changesetem 26. Nedejte se zmýlit tím, že to není 28. 28 je číslo revize, které měl changeset v hg1. Čísla revizí se obecně mohou lišit.

Při importu oprávky se automaticky aktualizuje lokální kopie a změna je automaticky zaznamenána (ledaže je zadán flag `–no-commit`).

Změny z nepříbuzných repozitářů nelze importovat.

Nástroj import/export lze použít, nemůžeme-li se k jinému repozitáři připojit přímo. Oprávky lze například poslat e-mailem.

Pamatujme si, že tyto oprávky obsahují informace o repozitáři, komentář, jméno uživatele, atd - což je více než u tradiční UNIX oprávky, o níž pohovoříme v odstavci Error: Reference source not found.

7.4 Svázat/Rozvázat

Příkazy `bundle` a `unbundle` jsou podobné příkazům export/import. Síla příkazu bundle (svázat) spočívá v možnosti zadat společné předky(flag `–base`), o nichž se ví, že v druhém repozitáři existují. Mercurial potom sám určí, které changesety jsou zapotřebí pro bezpečný přenos oprávky do druhého repozitáře.

```
$ cd ../hg1
$ hg log -r 24
changeset:   24:a836f48b2edb
user:        Thorbjorn Jemander
date:        Mon Dec 28 14:28:39 2009 +0100
summary:     Cleaned up module a.
$ hg bundle -r 24 --base 23 ../hg1-bundle
1 changesets found
$ cd ../hg2
$ hg unbundle ../hg1-bundle
adding changesets
adding manifests
adding file changes
added 1 changesets with 2 changes to 2 files (+1 heads)
(run 'hg heads' to see heads, 'hg merge' to merge)
$ hg glog
o  changeset:   27:a836f48b2edb
|  tag:         tip
|  parent:      24:58f714a8f24c
|  user:        Thorbjorn Jemander
|  date:        Mon Dec 28 14:28:39 2009 +0100
```

```
|  summary:     Cleaned up module a.
|
| @  changeset:   26:21fdaa10b77d
| |  user:        Thorbjorn Jemander
| |  date:        Mon Dec 28 15:39:30 2009 +0100
| |  summary:     adding p
...
$ hg merge
2 files updated, 0 files merged, 0 files removed, 0 files unresolved
(branch merge, don't forget to commit)
$ hg ci -m"Merged in bundle from hg1."
```

8 Práce s oprávkami

V tomto odstavci pohovoříme o různých způsobech práce s oprávkami. Důvodem pro tuto rozpravu je problém s *překážejícími nekomitovanými změnami* (OUC - outstanding uncommitted changes), on nichž jsem se již dříve zmínil. Tento problém se vhodně dá řešit právě oprávkami.

Problém OUC vznikne, když se snažíme v jednom souboru sloučit změny ze tří zdrojů: změny vzdáleného repozitáře, lokální komitované změny a lokální nekomitované změny.

Je to právě ten počet zdrojů, který způsobuje problém. Pokud jej dokážeme redukovat na dva, je to bezpečnější. Můžeme buď

1. Komitovat **před** sloučením (a tím odstranit lokální změny)
2. Nekomitovat **žádné** změny před sloučením (a tím nemít žádné komitované změny)

Někdy ovšem není při slučování zřejmé, že vznikne problém a můžeme mít potřebu jej řešit až po vzniku situace. Řešením je přesunout lokální změny do oprávky, provést sloučení a poté lokální změny znovu použít.

Zmíním se o dvou způsobech práce s oprávkami: *oprávky UNIX* a *fronty oprávek.* Oprávky UNIX nejsou konceptem Mercurialu, ale jsou jím dobře integrovány.

8.1 Oprávky UNIX

Vytvořme dva repozitáře a situaci OUC.

```
$ mkdir hg5
$ cd hg5
$ hg init
$ for i in {1..13}; do echo "row $i" >> a; done
$ cat a
row 1
row 2
row 3
row 4
row 5
row 6
row 7
row 8
row 9
row 10
row 11
row 12
row 13
$ hg addr
adding a
$ hg ci -m"added a"
$ cd ..
$ hg clone hg5 hg6
updating working directory
1 files updated, 0 files merged, 0 files removed,
0 files unresolved
$ cd hg5
$ gedit a
```

```
row 1
row 2
row 3 foo
row 4
row 5
row 6
row 7
row 8
row 9
row 10
row 11
row 12
row 13
```

The file hg5/a

Pomocí editoru přidáme řetězec "foo" k row 3 v souboru *hg5/a*.

```
row 1
row 2
row 3
row 4
row 5
row 6
row 7
row 8 bar
row 9
row 10
row 11
row 12
row 13
```

The file hg6/a (I)

Komitujeme změnu a přejdeme do hg6:

```
$ hg ci -m"added foo to row 3"

$ cd ../hg6
$ gedit a
```

Přidáme řetězec "bar" k row 8 v souboru *hg6/a* a komitujeme změnu.

```
$ hg ci -m"added bar to row 8"
$ gedit a
```

Opět editujeme soubor *hg6/a* a přidáme "local change" k row12 a nekomitujeme

```
row 1
row 2
row 3
row 4
row 5
row 6
row 7
row 8 bar
row 9
row 10
row 11
row 12 local change
row 13
```

The file hg6/a (II)

Nyní se pokusíme přitáhnout změnu z *hg5* a provést sloučení:

```
$ hg fetch
abort: outstanding uncommitted changes # (překážející nekomitované změny)
```

Odstraníme lokální změny vytvořením oprávky a provedením příkazu `revert`:

```
$ hg diff > /tmp/mypatch.diff          # vytvoření oprávky
$ hg revert -a     # volba -a vrací k předchozímu stavu všechny změny
reverting a                # které jsou zde náhodou jenom v a
$ hg fetch
pulling from /home/thorman/hg-tutorial/hg5
searching for changes
adding changesets
adding manifests
adding file changes
added 1 changesets with 1 changes to 1 files (+1 heads)
updating to 2:655659cc91c8
1 files updated, 0 files merged, 0 files removed, 0 files unresolved
merging with 1:66fffbe5c693
merging a
0 files updated, 1 files merged, 0 files removed, 0 files unresolved
new changeset 3:ab4970978198 merges remote changes with local
$ cat a
row 1
row 2
row 3 foo
row 4
row 5
row 6
row 7
row 8 bar
row 9
row 10
row 11
row 12
row 13
```

Vrátíme zpátky naše změny:

```
$ patch < /tmp/mypatch.diff
$ cat a                          # Prohlédněme si a nyní
row 1
row 2
row 3 foo
row 4
row 5
row 6
row 7
row 8 bar
row 9
row 10
row 11
row 12 local change
row 13
$ hg diff                        # Kontrolujeme lokální změny...
diff -r ab4970978198 a
--- a/a   Wed Dec 30 08:32:05 2009 +0100
+++ b/a   Wed Dec 30 08:35:45 2009 +0100
@@ -9,5 +9,5 @@
 row 9
 row 10
 row 11
-row 12
+row 12 local change      # Změny nejsou komitovány, OK.
 row 13
```

Hotovo. Generovali jsme diff příkazem `hg diff` a potom použili UNIXový příkaz `patch` pro vrácení změn. Následují *fronty oprávek*, což je systematičtější způsob práce s oprávkami.

8.2 Fronty oprávek

Abychom mohli pracovat s frontami oprávek, musíme povolit extenzi *mq*:

```
[ui]
username = Thorbjorn Jemander

[extensions]
graphlog=
hgext.extdiff =
fetch=
hgext.mq=

[extdiff]
cmd.kdiff3 =

[merge-tools]
kdiff3.args = $base $local $other -o $output
```

The file ~/.hgrc

Dále si ukážeme řešení výše zavedeného problému pomocí front oprávek. Spustíme stejnou situaci změnou souboru *hg5/a* a pokusem stáhnout změny do *hg6*:

```
$ cd ../hg5
$ echo "row 14" >> a
$ hg ci -m"added row 14"
$ cd ../hg6
$ hg fetch
abort: outstanding uncommitted changes
```

8.2.1 qinit

Pro práci s frontami oprávek potřebujeme nejprve inicializovat repozitář:

```
$ hg qinit
```

8.2.2 qnew

Novou oprávku vytvoříme příkazem `qnew`. Flag `-f` vybere lokální změny pracovní kopie a vytvoří z nich oprávku .
My si takto vytvoříme oprávku s názvem "myfeature".

```
$ hg qnew -f myfeature
```

8.2.3 qseries a qapplied

Jsou dvě sady oprávek, s nimiž pracujeme: sada všech oprávek a sada použitých oprávek. Příkaz `qseries` ukáže sadu všech oprávek a `qapplied` sadu použitých oprávek:

```
$ hg qseries
myfeature                          # Podtržení znamená applied (použité).
$ hg qapplied
myfeature
$ hg st
$ hg log
changeset:   4:5fbc699a1bb1
tag:         qtip
tag:         tip
tag:         myfeature
tag:         qbase
user:        Thorbjorn Jemander
date:        Thu Dec 31 11:03:05 2009 +0100
summary:     [mq]: myfeature

changeset:   3:ab4970978198
tag:         qparent
parent:      2:655659cc91c8
parent:      1:66fffbe5c693
user:        Thorbjorn Jemander
date:        Wed Dec 30 08:32:05 2009 +0100
summary:     Automated merge with file:///home/thorman/hg-tutorial/hg5
...
```

Příkaz `qnew` tedy vybral lokální změny a vytvořil z nich oprávku "myfeature", jež je nyní aplikována jako changeset 4.

8.2.4 qpop

Neměli jsme odstranit oprávku? Ano, uděláme to nyní příkazem `qpop`:

```
$ hg qpop
patch queue now empty        # (fronta oprávek je nyní prázdná)
$ hg qseries
myfeature
$ hg qapplied
$ hg log
changeset:   3:ab4970978198
tag:         qparent
parent:      2:655659cc91c8
parent:      1:66fffbe5c693
user:        Thorbjorn Jemander
date:        Wed Dec 30 08:32:05 2009 +0100
summary:     Automated merge with file:///home/thorman/hg-tutorial/hg5
...
```

Oprávku jsme vystrčili (popped), takže není ve hře a můžeme přitáhnout další obsah:

```
$ hg fetch ../hg5
pulling from ../hg5
searching for changes
adding changesets
adding manifests
adding file changes
added 1 changesets with 1 changes to 1 files (+1 heads)
updating to 4:05b27b8704b0
1 files updated, 0 files merged, 0 files removed, 0 files unresolved
merging with 3:ab4970978198
merging a
0 files updated, 1 files merged, 0 files removed, 0 files unresolved
new changeset 5:6209c396169f merges remote changes with local
```

Nyní můžeme naši oprávku opět importovat:

```
$ hg import --no-commit .hg/patches/myfeature
applying .hg/patches/myfeature
$ hg diff
diff -r 6209c396169f a
--- a/a   Thu Dec 31 11:05:33 2009 +0100
+++ b/a   Thu Dec 31 11:08:25 2009 +0100
@@ -9,6 +9,6 @@
 row 9
 row 10
 row 11
-row 12
+row 12 local change
 row 13
 row 14
```

Jsme ve stejné situaci jako po použití unixové oprávky výše: naše lokální změny nejsou komitovány a proto sídlí pouze v naší pracovní kopii (v důsledku použití flagu `--no-commit`).

Toto však není přesně ten způsob, jakým bychom měli pracovat s oprávkami. Použití příkazu `import` není zrovna elegantní, protože vyžaduje znalost, že oprávky jsou uloženy v *.hg/patches* (což se může změnit) a frontu oprávek vlastně obcházíme.

Nyní pro řešení našeho problému použijeme frontu oprávek správnějším způsobem.

8.2.5 qpush

Nejprve opět vytvoříme problém *OUC* a smažeme starou oprávku:

```
$ cd ../hg5
$ echo "row 15" >> a
$ hg ci -m"added row 15"
$ cd ../hg6
$ hg qdelete myfeature            # delete the patch from the example above.
$ hg fetch
abort: outstanding uncommitted changes
```

Přeneseme lokální změny do oprávky, vysuneme ji mimo hru, přitáhneme změny a oprávku vrátíme zpět:

```
$ hg qnew -f myfeature2
$ hg qpop
patch queue now empty
$ hg fetch
pulling from /home/thorman/hg-tutorial/hg5
searching for changes
adding changesets
adding manifests
adding file changes
added 1 changesets with 1 changes to 1 files (+1 heads)
updating to 6:868c5031ed0b
1 files updated, 0 files merged, 0 files removed, 0 files unresolved
merging with 5:6209c396169f
merging a
0 files updated, 1 files merged, 0 files removed, 0 files unresolved
new changeset 7:d44050773789 merges remote changes with local
$ hg qpush
applying myfeature2
now at: myfeature2
$ hg qapplied
myfeature2
$ hg log
changeset:   8:58197e6ac8e3
tag:         qtip
tag:         myfeature2
tag:         tip
tag:         qbase
user:        Thorbjorn Jemander
date:        Thu Dec 31 11:25:00 2009 +0100
summary:     imported patch myfeature2
...
```

Takže správné řešení problému *OUC* je: `qnew`, `qpop`, `fetch` a `qpush`.

Rozdíl mezi metodou `import` a metodou `qpush` je ten, že nyní je oprávka zpátky a použitá

(takže je na vrcholu záznamu). Někdo by mohl namítat, že to není to, co bychom měli chtít, protože lokální změny nejsou připraveny pro záznam. Bez obav, v dalším si ukážeme práci s oprávkami s pěkným komitem na konci.

8.2.6 qrefresh a qdiff

Řekněme, že se svými lokálními změnami nejsme hotovi. Žádný problém. Změny můžeme přidat do oprávky:

```
$ echo "row 16" >> a
$ hg qrefresh                        # Lokální změny přidány do oprávky.
$ hg diff                            # Žádné překážející změny,
$ hg qdiff                           # jsou všechny v oprávce:
diff -r d44050773789 a
--- a/a   Thu Dec 31 11:24:44 2009 +0100
+++ b/a   Thu Dec 31 12:16:58 2009 +0100
@@ -9,7 +9,8 @@
 row 9
 row 10
 row 11
-row 12
+row 12 local change
 row 13
 row 14
 row 15
+row 16
```

Příkaz `qdiff` ukazuje změny aktuální oprávky. Existují-li také lokální změny, ukazuje `qdiff` lokální změny plus změny v aktuální oprávce (po příkazu `qrefresh`).

Příkaz `qrefresh` absorbuje lokální změny do aktuální oprávky. Oprávku jsme však dosud nepředali do záznamu historie v repozitáři. Uděláme to za chvilku, nejprve si vytvoříme několik dalších oprávek.

8.2.7 Více oprávek

Aktuální oprávku lze rozšiřovat o další změny, ale lze také vytvořit oprávku novou. Může být, že aktuální oprávka je poměrně stálá, ale my chceme provést nějaké experimentální změny a nejsme si jisti, že je budeme chtít uchovat:

```
$ hg qnew experimental
$ echo "row 17: experimental" >> a
$ hg diff
diff -r 72ed90212e43 a
--- a/a   Thu Dec 31 11:52:53 2009 +0100
+++ b/a   Thu Dec 31 11:53:02 2009 +0100
@@ -14,3 +14,4 @@
 row 14
 row 15
 row 16
+row 17: experimental
$ hg qrefresh                            # Absorbovat lokální změny.
$ hg diff                                # Žádné změny pracovní kopie.
$ hg qdiff                               # Změny jsou v oprávce:
```

```
diff -r ab51ff1f1b56 a
--- a/a   Thu Dec 31 11:34:13 2009 +0100
+++ b/a   Thu Dec 31 11:53:05 2009 +0100
@@ -14,3 +14,4 @@
 row 14
 row 15
 row 16
+row 17: experimental
```

8.2.8 Pohyb mezi oprávkami: qgoto

Dosud jsme se mezi oprávkami pohybovali použitím `qpop` a `qpush`. Existuje také `qgoto`:

```
$ hg qseries
myfeature2
experimental
$ hg qapplied
myfeature2
experimental
$ hg qpop                                   # Odebrat posledně použitou oprávku
now at: myfeature2
$ hg qapplied
myfeature2
$ cat a
...
row 11
row 12 local change
row 13
row 14
row 15
row 16                                      # Žádná experimentální řada 17
$ hg qpop
patch queue now empty
$ cat a
...
row 11
row 12                                      # Žádné "lokální změny"
row 13
row 14
row 15                                      # Žádná řada 16 (v myfeature2).
$ hg qgoto experimental                     # qgoto použije/odebere oprávky dle
applying myfeature2                         # konkrétní potřeby.
applying experimental
now at: experimental
```

8.2.9 Zápis oprávek: qfold and qfinish

Když si prohlédneme záznam, vidíme, že použité oprávky jsou zobrazeny jako řádné changesety na vrcholu historie s přidanými tagy (`qbase, qtip)`, názvy oprávek a s generickými komentáři, začínajícími slovy "imported patch":

```
$ hg log
changeset:   9:b3a51e5dab2e
tag:         qtip
tag:         tip
tag:         experimental
user:        Thorbjorn Jemander
```

```
date:        Thu Dec 31 12:00:02 2009 +0100
summary:     imported patch experimental

changeset:   8:4a97313129b4
tag:         qbase
tag:         myfeature2
user:        Thorbjorn Jemander
date:        Thu Dec 31 12:00:02 2009 +0100
summary:     imported patch myfeature2

changeset:   7:d44050773789
tag:         qparent
parent:      6:868c5031ed0b
parent:      5:6209c396169f
user:        Thorbjorn Jemander
date:        Thu Dec 31 11:24:44 2009 +0100
summary:     Automated merge with file:///home/thorman/hg-tutorial/hg5
...
```

Takto si ale zřejmě výsledný vzhled našeho repozitáře nepředstavujeme. Chtěli bychom použít vlastní komentáře, možná bychom chtěli jediný changeset a také mít možnost prohlédnout si změny ještě před komitem – i když chceme předat (commit) několik oprávek v jednom changesetu.

Než tak učiníme, vytvoříme třetí oprávku *experimental2*, a potom se přesuneme k oprávce *experimental*:

```
$ hg qnew experimental2
$ echo b > b
$ hg add b                  # Ano, oprávky "umí" add/remove/addremove/rename
$ hg qrefresh
$ hg diff
$ hg qgoto myfeature2
popping experimental2
popping experimental
now at: myfeature2
$ hg qseries
myfeature2
experimental
experimental2
```

Nyní tedy nadešel čas odevzdat *myfeature2* a *experimental* repozitáři jako jediný changeset. Nejprve sbalíme (fold) obě oprávky dohromady:

```
$ hg qfold experimental
$ hg qseries
myfeature2          # Oprávka experimental je pryč, vnořena do myfeature2.
experimental2
$ hg log
changeset:   8:cff35e37e52a
tag:         qtip
tag:         myfeature2
tag:         tip
tag:         qbase
user:        Thorbjorn Jemander
date:        Thu Dec 31 12:46:26 2009 +0100
```

```
summary:      [mq]: myfeature2
...
```

Uf. Zapomněli jsme použít flag `-m` flag v příkazu `qfold` takže máme stále ten nudný komentář "[mq]: myfeature2". Nezoufejme, komentář komitu můžeme změnit příkazem `qrefresh`:

```
$ hg qrefresh -m"Implemented my feature no 2"
$ hg head
changeset:    8:42905ba00dba
tag:          qtip
tag:          myfeature2
tag:          tip
tag:          qbase
user:         Thorbjorn Jemander
date:         Thu Dec 31 12:52:49 2009 +0100
summary:      Implemented my feature no 2
```

To je lepší. Než provedeme komit, můžeme si změny prohlédnout:

```
$ hg qdiff
diff -r d44050773789 a
--- a/a   Thu Dec 31 11:24:44 2009 +0100
+++ b/a   Thu Dec 31 12:53:45 2009 +0100
@@ -9,7 +9,9 @@
 row 9
 row 10
 row 11
-row 12
+row 12 local change
 row 13
 row 14
 row 15
+row 16
+row 17: experimental
```

Výborně, výsledná oprávka obsahuje změny jak z *myfeature2,* tak z *experimental*. Oprávku předáme s použitím příkazu `qfinish`:

```
$ hg qfinish 8                # 8 = revize 8.
$ hg log
changeset:    8:42905ba00dba
tag:          tip
user:         Thorbjorn Jemander
date:         Thu Dec 31 12:52:49 2009 +0100
summary:      Implemented my feature no 2
...
```

Musíte zadat číslo revize, nebo „tip“, nebo „-a“, chcete-li předat všechny použité oprávky.

Jsme hotovi: máme jediný changeset obsahující všechny naše změny

Pracovat s frontou oprávek mi přijde velmi vhodné. Když ovšem narazíme na konflikt, je jejich řešení poněkud odlišné od postupu výše popsaného. Může se stát, že je naše oprávka *odmítnuta*.

8.3 Odmítnuté oprávky

Vytvořme situaci, ve které selže aplikace oprávky:

```
$ gedit a
```

Provedeme dvě změny v souboru *a*, jednu v řadě 2 a druhou v řadě 14, podle zobrazení vpravo. Vytvoříme unixovou oprávku:

```
$ hg diff > /tmp/mypatch2.diff
```

Nyní změny odvoláme a provedeme novou v řadě 14 dle obrázku vpravo. Soubor uložíme a použijeme `mypatch2`:

```
$ hg revert a
$ gedit a
$ patch < /tmp/mypatch2.diff
patching file a
Hunk #2 FAILED at 11.
1 out of 2 hunks FAILED -- saving rejects to file a.rej
```

```
row 1
row 2 change #1
row 3 foo
row 4
row 5
row 6
row 7
row 8 bar
row 9
row 10
row 11
row 12 local change
row 13
row 14 change #2
row 15
row 16
```

```
row 1
row 2
row 3 foo
row 4
row 5
row 6
row 7
row 8 bar
row 9
row 10
row 11
row 12 local change
row 13
row 14 other change
row 15
row 16
row 17: experimental
```

The file hg6/a (II)

Použití oprávky z nějakého důvodu selhalo. Co jsou ty porce (hunks), na něž se zdůvodnění odvolává? Prohlédneme si soubor oprávky:

```
$ cat /tmp/mypatch2.diff
diff -r 6b75f218fcc1 a
--- a/a Fri Jan 08 17:38:15 2010 +0100
+++ b/a Fri Jan 08 17:52:03 2010 +0100
@@ -1,5 +1,5 @@          ← Line number information
 row 1
-row 2
+row 2 change #1          Hunk(porce) #1
 row 3 foo
 row 4
 row 5
@@ -11,7 +11,7 @@          ← Line number information
 row 11
 row 12 local change
 row 13
-row 14                   Hunk(porce) #2
+row 14 change #2
 row 15
 row 16
 row 17: experimental
```

Soubor oprávky je sekvence porcí. Každá porce ve výše uvedeném příkladu obsahuje informaci o číslech řádků a kontextovou informaci. Kontextová informace je několik řádků před a po samotné změně, ukazující vzhled souboru při generování diffu.

Když příkaz *patch* aplikuje oprávku, nalezne místo pro vytvoření změny s použitím informace jak o

kontextu, tak o číslech řádku. To umožňuje přemisťování kódu a přitom lze vytvářet oprávky ze souborů, jež nejsou totožné se souborem původně srovnáváným (diffed).

Když ovšem příkaz *patch* nenalezne místo v souboru shodující se s kontextem porce, neví co má dělat. V našem případě jsme lokálně změnili řadu 14, takže to *patch* vzdává a *odmíná porci*. Neúspěšní kandidáti jsou odloženi do souboru "<file>.rej" files; v našem případě to byl soubor *a.rej*. Prohlédneme si jej:

```
$ cat a.rej
***************
*** 11,17 ****
  row 11
  row 12 local change
  row 13
- row 14
  row 15
  row 16
  row 17: experimental
--- 11,17 ----
  row 11
  row 12 local change
  row 13
+ row 14 change #2
  row 15
  row 16
  row 17: experimental
```

Můžeme zde vidět, co se zkoušelo provést: odstranit "row 14" a přidat "row 14 change #2". Tato informace nám usnadňuje analýzu situace a rozhodnutí, co učinit. V tomto specielním případě jde o to, zda má řada 14 obsahovat to, co je naznačováno, nebo by měla obsahovat "jinou změnu", kterou někdo chce. To vyžaduje osobní asistenci.

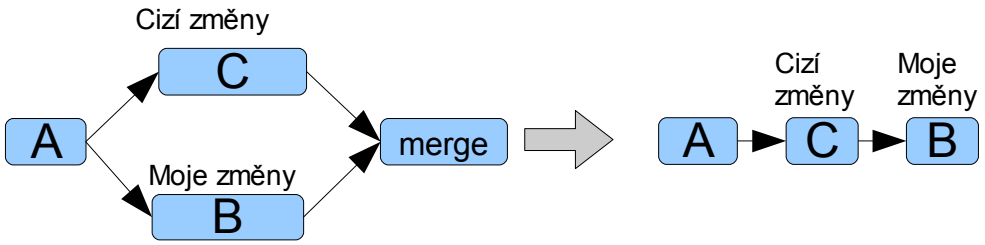
Když se vyskytne selhání oprávky, tak jednoduše otevřeme soubor "<file>.rej", podíváme se o co se *patch* snažil a potom změníme odpovídající soubor.

Odmítnutí oprávky se může vyskytnout nejenom při práci s unixovými oprávkami, ale také když pracujeme s jinými oprávkovými nástroji. Máme-li povolenou extenzi pro práci s oprávkami, můžeme očekávat spíše odmítnutí oprávky než obyčejný konflikt. Pro řešení odmítnutých oprávek však nemáme žádný vhodný grafický nástroj jako pro řešení obyčejných konfliktů.

9 Další extenze

9.1 Rebase

Kapitola pojednává o napřímení historie a umístění našich změn na konec. Rebase = přeskupit.



Obvykle, když pracujeme souběžně s jinými lidmi, se dostaneme do situace, zobrazené na levé straně obrázku: změny jiných lidí (C) se objeví souběžně s našimi změnami (B). Přeskupením změn můžeme přeměnit situaci k obrazu v pravé části ilustrace a odstranit potřebu sloučení.

Extenzi povolíme přidáním `rebase=` do našeho souboru ~/.hgrc:

```
[ui]
username = Thorbjorn Jemander

[extensions]
graphlog=
hgext.extdiff =
fetch=
hgext.mq=
rebase=

[extdiff]
cmd.kdiff3 =

[merge-tools]
kdiff3.args = $base $local $other -o $output
```

The file ~/.hgrc

Vytvoříme nějakou historii, naklonujeme repozitář, provedeme lokální změny, zapíšeme je do repo a další změny si stáhneme:

```
$ mkdir hg7
$ cd hg7
$ hg init
$ echo a > a; hg add a; hg ci -ma
$ cd ..
$ hg clone hg7 hg8
updating to branch default
1 files updated, 0 files merged, 0 files removed, 0 files unresolved
$ cd hg8
```

```
$ echo b > b; hg add b; hg ci -mb
$ cd ../hg7
$ echo c > c; hg add c; hg ci -mc
$ cd ../hg8
$ hg pull
pulling from /home/thorman/hg-tutorial/hg7
searching for changes
adding changesets
adding manifests
adding file changes
added 1 changesets with 1 changes to 1 files (+1 heads)
(run 'hg heads' to see heads, 'hg merge' to merge)
$ hg merge
1 files updated, 0 files merged, 0 files removed, 0 files unresolved
(branch merge, don't forget to commit)
$ hg ci -m"merged"
$ hg glog
@    changeset:   3:ab66109a53ab
|\   tag:         tip
| |  parent:      1:9fe6f317436f
| |  parent:      2:875c5bcfa0a2
| |  user:        Thorbjorn Jemander
| |  date:        Fri Jan 01 09:46:14 2010 +0100
| |  summary:     merged b c
| |
| o  changeset:   2:875c5bcfa0a2
| |  parent:      0:c054b2eb6a95
| |  user:        Thorbjorn Jemander
| |  date:        Fri Jan 01 09:43:14 2010 +0100
| |  summary:     c
| |
o |  changeset:   1:9fe6f317436f
|/   user:        Thorbjorn Jemander
|    date:        Fri Jan 01 09:42:58 2010 +0100
|    summary:     b
|
o  changeset:   0:c054b2eb6a95
   user:        Thorbjorn Jemander
   date:        Fri Jan 01 09:42:19 2010 +0100
   summary:     a
```

Máme nyní paralelní historii, vyžadující sloučení. Přeji si, aby moje změny byly poslední. S pomocí příkazu `rebase` mohu změny ve stromové struktuře přesunovat. Existuje *zdroj* a *destinace*. V tomto případě chci přesunout svoji změnu *b* (changeset 1), která je zdrojem za vzdálenou změnu *c* (changeset 2), což je destinace:

```
$ rebase -s 1 -d 2                      # -s = --zdroj(source), -d = --dest
saving bundle to /home/thorman/hg-tutorial/hg8/.hg/strip-backup/9fe6f317436f-
temp
adding branch
adding changesets
adding manifests
adding file changes
added 2 changesets with 2 changes to 2 files
rebase completed
$ hg glog
```

```
@  changeset:   2:35252c967029
|  tag:         tip
|  user:        Thorbjorn Jemander
|  date:        Fri Jan 01 09:42:58 2010 +0100
|  summary:     b
|
o  changeset:   1:875c5bcfa0a2
|  user:        Thorbjorn Jemander
|  date:        Fri Jan 01 09:43:14 2010 +0100
|  summary:     c
|
o  changeset:   0:c054b2eb6a95
   user:        Thorbjorn Jemander
   date:        Fri Jan 01 09:42:19 2010 +0100
   summary:     a
```

Není to nádhera? Všimněme si, jak sloučení magicky zmizelo. Potřeba sloučení zmizela, jakmile jsme založili *b* na *c*, místo na *a*.

Tato extenze také poskytuje další volbu `--rebase` pro příkaz pull, která je šikovná. Vyzkoušíme si ji:

```
$ cd ../hg7
$ echo d > d
$ hg add d
$ hg ci -md
$ cd ../hg8
$ hg pull --rebase
pulling from /home/thorman/hg-tutorial/hg7
searching for changes
adding changesets
adding manifests
adding file changes
added 1 changesets with 1 changes to 1 files (+1 heads)
(run 'hg heads' to see heads, 'hg merge' to merge)
saving bundle to /home/thorman/hg-tutorial/hg8/.hg/strip-backup/35252c967029-
temp
adding branch
adding changesets
adding manifests
adding file changes
added 2 changesets with 2 changes to 2 files
rebase completed
$ hg glog
@  changeset:   3:3591d86d1dfd
|  tag:         tip
|  user:        Thorbjorn Jemander
|  date:        Fri Jan 01 09:42:58 2010 +0100
|  summary:     b
|
o  changeset:   2:06b50be3693a
|  user:        Thorbjorn Jemander
|  date:        Fri Jan 01 09:54:24 2010 +0100
|  summary:     d
|
o  changeset:   1:875c5bcfa0a2
|  user:        Thorbjorn Jemander
```

```
|  date:        Fri Jan 01 09:43:14 2010 +0100
|  summary:     c
|
o  changeset:   0:c054b2eb6a95
   user:        Thorbjorn Jemander
   date:        Fri Jan 01 09:42:19 2010 +0100
   summary:     a
```

Nyní se automaticky přesunulo *d* před naši změnu *b*, takže je *b* stále v historii poslední.

O přeskupování lze více číst zde: http://mercurial.selenic.com/wiki/RebaseProject

9.2 Další berle antiperle

9.2.1 Backout

Vezmeme si opět pracovní adresář *hg8* a budeme chtít zrušit či odvolat (back out) revizi 1 (c), jež není *tipem,* neboli není v historii poslední. Můžeme provést následující:

```
$ cd hg8
$ hg backout -r 1 -m"Backed out c"
removing c
created new head
changeset 4:5d9c49b4e432 backs out changeset 1:875c5bcfa0a2
the backout changeset is a new head - do not forget to merge
(use "backout --merge" if you want to auto-merge)
$ hg glog
@  changeset:   4:894e428a2855
|  tag:         tip
|  parent:      1:875c5bcfa0a2
|  user:        Thorbjorn Jemander
|  date:        Fri Jan 01 22:53:23 2010 +0100
|  summary:     Backed out c
|
| o  changeset:   3:3591d86d1dfd
| |  user:        Thorbjorn Jemander
| |  date:        Fri Jan 01 09:42:58 2010 +0100
| |  summary:     b
| |
| o  changeset:   2:06b50be3693a
|/   user:        Thorbjorn Jemander
|    date:        Fri Jan 01 09:54:24 2010 +0100
|    summary:     d
|
o  changeset:   1:875c5bcfa0a2
|  user:        Thorbjorn Jemander
|  date:        Fri Jan 01 09:43:14 2010 +0100
|  summary:     c
|
o  changeset:   0:c054b2eb6a95
   user:        Thorbjorn Jemander
   date:        Fri Jan 01 09:42:19 2010 +0100
   summary:     a
$ hg merge
2 files updated, 0 files merged, 0 files removed, 0 files unresolved
(branch merge, don't forget to commit)
$ hg ci -m"merged in c removal""
```

```
$ ls
a b d                                   # No c.
```

Odstranili jsme *c* použitím "negativní oprávky" hned za odstraňovaným changesetem. (Proto se vytvořilo další čelo.)

Pracujeme-li s frontou oprávek, můžeme to provést bez dalšího čela, pokud je changeset, který hodláme odstranit, ve skutečnosti oprávka. Jako cvičení lze vyzkoušet tento postup: *Just un-apply it by popping patches, delete it, and push back the patches you popped.*

9.2.2 Klon

Někdy se věci tak pokazí, že nám nepomůže ani revert, rollback nebo backout. Můžeme zjednat nápravu ještě tak, že provedeme klon jisté části repozitáře a starou část postupně nahradíme novou. Předpokládejme, že všechno za changesetem 2 (*d*) je špatné:

```
$ hg glog
@    changeset:   5:607d22c7c3c8
|\   tag:         tip
| |  parent:      4:894e428a2855
| |  parent:      3:3591d86d1dfd
| |  user:        Thorbjorn Jemander
| |  date:        Fri Jan 01 22:55:32 2010 +0100
| |  summary:     merged in c removal
| |
| o  changeset:   4:894e428a2855
| |  parent:      1:875c5bcfa0a2
| |  user:        Thorbjorn Jemander
| |  date:        Fri Jan 01 22:53:23 2010 +0100
| |  summary:     Backed out c
| |
o |  changeset:   3:3591d86d1dfd
| |  user:        Thorbjorn Jemander
| |  date:        Fri Jan 01 09:42:58 2010 +0100
| |  summary:     b
| |
o |  changeset:   2:06b50be3693a
|/   user:        Thorbjorn Jemander
|    date:        Fri Jan 01 09:54:24 2010 +0100
|    summary:     d
|
o  changeset:   1:875c5bcfa0a2
|  user:        Thorbjorn Jemander
|  date:        Fri Jan 01 09:43:14 2010 +0100
|  summary:     c
|
o  changeset:   0:c054b2eb6a95
   user:        Thorbjorn Jemander
   date:        Fri Jan 01 09:42:19 2010 +0100
   summary:     a
$ cd ..
$ hg clone -r 2 hg8 hg8-new
requesting all changes
adding changesets
adding manifests
```

```
adding file changes
added 3 changesets with 3 changes to 3 files
updating to branch default
3 files updated, 0 files merged, 0 files removed, 0 files unresolved

$ mv hg8 hg8-old                    # postupně adresáře přejmenujeme
$ mv hg8-new hg8
$ cd hg8
$ hg glog
@  changeset:   2:06b50be3693a
|  tag:         tip
|  user:        Thorbjorn Jemander
|  date:        Fri Jan 01 09:54:24 2010 +0100
|  summary:     d
|
o  changeset:   1:875c5bcfa0a2
|  user:        Thorbjorn Jemander
|  date:        Fri Jan 01 09:43:14 2010 +0100
|  summary:     c
|
o  changeset:   0:c054b2eb6a95
   user:        Thorbjorn Jemander
   date:        Fri Jan 01 09:42:19 2010 +0100
   summary:     a
$ cat .hg/hgrc
[paths]
default = /home/thorman/hg-tutorial/hg8
```

Protože jsme provedli klon, odkazuje se nová kopie na svého rodiče, čímž po přejmenování odkazuje sama na sebe.

Potřebujeme aktualizovat implicitní cestu, aby odpovídala novému repozitáři.

Varovné slovo: když klonujeme pracovní adresář, kopíruje se jenom zaznamenaná historie. Změny v pracovní kopii se nezachovají a totéž platí pro informaci o frontě oprávek. Tato se ztratí během procesu klonování a použité oprávky se změní v regulérní changesety. Nepoužité oprávky jsou úplně zapomenuty a musejí být přeneseny do nového repozitáře ručně, pokud o ně stojíme.

9.2.3 Strip

Pokud jsme si nainstalovali extenzi *mq*, máme přístup k příkazu `strip`, jímž lze rychle odstřihnout libovolnou část historie. Vezměme si repozitář *hg3*, který jsme použili v příkladu "nepříbuzné repozitáře":

```
$ cd hg3
$ hg up -C                          # Pouze mažeme lokální změny..
$ echo c3 >> C
$ hg ci -m"version 3 of c."
$ hg glog
@  changeset:   2:e66fd480701c
|  tag:         tip
|  user:        Thorbjorn Jemander
|  date:        Mon Dec 28 17:10:32 2009 +0100
|  summary:     version 3 of c.
|
o  changeset:   1:3e236b6d437f
|  user:        Thorbjorn Jemander
|  date:        Mon Dec 28 15:05:15 2009 +0100
```

```
|  summary:     adding c
|
o  changeset:   0:1f36da6d72b9
   user:        Thorbjorn Jemander
   date:        Mon Dec 28 15:03:21 2009 +0100
   summary:     adding a
$ hg strip 1
0 files updated, 0 files merged, 1 files removed, 0 files unresolved
saving bundle to /home/thorman/hg-tutorial/hg3/.hg/strip-backup/3e236b6d437f-
backup
$ hg glog
@  changeset:   0:1f36da6d72b9
   tag:         tip
   user:        Thorbjorn Jemander
   date:        Mon Dec 28 15:03:21 2009 +0100
   summary:     adding a
```

Příkaz `strip` odstranil určenou revizi a všechny její potomky. Vězme, že tím neodstraníme všechny pozdější revize, pouze ty, jímž je určená revize předkem. Lze takto například odstřihnout nežádaná čela. Vyzkoušíme si to po vytvoření dvou čel:

```
$ echo a2 >> a; hg ci -ma2
$ echo a3 >> a; hg ci -ma3
$ echo a4 >> a; hg ci -ma4
$ hg up 1
$ echo A1 >> a; hg ci -mA1
created new head
$ echo A2 >> a; hg ci -mA2
$ hg glog
@  changeset:   5:6464ba3236fe
|  tag:         tip
|  user:        Thorbjorn Jemander
|  date:        Sat Jan 09 07:36:47 2010 +0100
|  summary:     A2
|
o  changeset:   4:4a63d6f5a351
|  parent:      1:990b75b71265
|  user:        Thorbjorn Jemander
|  date:        Sat Jan 09 07:36:29 2010 +0100
|  summary:     A1
|
| o  changeset:   3:07003b668c27
| |  user:        Thorbjorn Jemander
| |  date:        Sat Jan 09 07:36:21 2010 +0100
| |  summary:     a4
| |
| o  changeset:   2:a9ea0e42df43
|/   user:        Thorbjorn Jemander
|    date:        Sat Jan 09 07:36:17 2010 +0100
|    summary:     a3
|
o  changeset:   1:990b75b71265
|  user:        Thorbjorn Jemander
|  date:        Sat Jan 09 07:36:13 2010 +0100
|  summary:     a2
```

```
|
o  changeset:   0:1f36da6d72b9
   user:        Thorbjorn Jemander
   date:        Mon Dec 28 15:03:21 2009 +0100
   summary:     adding a
```

Dejme tomu, že jsme změnili názor a nechceme změny *a3* a *a4*, protože nechceme slučovat. Mohli bychom je vymazat příkazem *backout,* ale to bychom slučovat stejně museli, takže to není příliš praktické. Spasí nás příkaz `strip`. Stačí říci, že chceme odtrhnout revizi 2 a její potomky:

```
$ hg strip 2
saving bundle to /home/thorman/hg-tutorial/chapter-repos/hg3-9.2.3/.hg/strip-
backup/a9ea0e42df43-backup
saving bundle to /home/thorman/hg-tutorial/chapter-repos/hg3-9.2.3/.hg/strip-
backup/a9ea0e42df43-temp
adding branch
adding changesets
adding manifests
adding file changes
added 2 changesets with 2 changes to 1 files
$ hg glog
@  changeset:   3:6464ba3236fe
|  tag:         tip
|  user:        Thorbjorn Jemander
|  date:        Sat Jan 09 07:36:47 2010 +0100
|  summary:     A2
|
o  changeset:   2:4a63d6f5a351
|  user:        Thorbjorn Jemander
|  date:        Sat Jan 09 07:36:29 2010 +0100
|  summary:     A1
|
o  changeset:   1:990b75b71265
|  user:        Thorbjorn Jemander
|  date:        Sat Jan 09 07:36:13 2010 +0100
|  summary:     a2
|
o  changeset:   0:1f36da6d72b9
   user:        Thorbjorn Jemander
   date:        Mon Dec 28 15:03:21 2009 +0100
   summary:     adding a
```

Inkriminované změny změny zmizely, ale pozdější změny *A1* a *A2* zůstaly. Všimneme si také, že došlo k automatickému přečíslování revizí.

9.3 Transplantace, neboli vybírání třešniček

Tuto extenzi povolíme přidáním řádku *transplant=* do souboru ~/.hgrc:

```
[ui]
username = Thorbjorn Jemander

[extensions]
graphlog=
hgext.extdiff =
fetch=
hgext.mq=
rebase=
transplant=

[extdiff]
cmd.kdiff3 =

[merge-tools]
kdiff3.args = $base $local $other -o $output
```

The file ~/.hgrc

Vytvoříme větev *b1*, obsahující změny *b* a *c*:

```
$ mkdir hg9; cd hg9; hg init
$ echo a > a; hg add a; hg ci -ma
$ hg branch b1
marked working directory as branch b1
$ echo b > b; hg add b; hg ci -mb
$ echo c > c; hg add c; hg ci -mc
$ hg up default
$ echo d > d; hg add d; hg ci -md
created new head
$ hg glog
@  changeset:   3:8119abef4e08
|  tag:         tip
|  parent:      0:b1c552afb583
|  user:        Thorbjorn Jemander
|  date:        Sat Jan 02 06:36:58 2010 +0100
|  summary:     d
|
| o  changeset:   2:67cada58a4ba
| |  branch:      b1
| |  user:        Thorbjorn Jemander
| |  date:        Sat Jan 02 06:36:49 2010 +0100
| |  summary:     c
| |
| o  changeset:   1:7f057f35689a
|/   branch:      b1
|    user:        Thorbjorn Jemander
|    date:        Sat Jan 02 06:36:45 2010 +0100
|    summary:     b
|
o  changeset:   0:b1c552afb583
   user:        Thorbjorn Jemander
   date:        Sat Jan 02 06:33:12 2010 +0100
   summary:     a
```

Jak bychom postupovali, abychom přenesli změny z větve *b1* do větve implicitní (defaultní)? Provedli bychom sloučení. Co když ale chceme přenést pouze changeset 1 (*b)?* Použijeme příkaz `transplant`:

```
$ hg transplant -b b1 1
applying 7f057f35689a
7f057f35689a transplanted to 35ec31cb6706
$ hg glog
@  changeset:   4:35ec31cb6706
|  tag:         tip
|  user:        Thorbjorn Jemander
|  date:        Sat Jan 02 06:36:45 2010 +0100
|  summary:     b
|
o  changeset:   3:8119abef4e08
|  parent:      0:b1c552afb583
|  user:        Thorbjorn Jemander
|  date:        Sat Jan 02 06:36:58 2010 +0100
|  summary:     d
|
| o  changeset:   2:67cada58a4ba
| |  branch:      b1
| |  user:        Thorbjorn Jemander
| |  date:        Sat Jan 02 06:36:49 2010 +0100
| |  summary:     c
| |
| o  changeset:   1:7f057f35689a
|/   branch:      b1
|    user:        Thorbjorn Jemander
|    date:        Sat Jan 02 06:36:45 2010 +0100
|    summary:     b
|
o  changeset:   0:b1c552afb583
   user:        Thorbjorn Jemander
   date:        Sat Jan 02 06:33:12 2010 +0100
   summary:     a
```

Můžeme také vybrat určité revize z jiného repozitáře použitím přepínače `-s` místo přepínače `-b`.

9.4 Založení serveru

Přejdeme do pracovního adresáře, který chceme sdílet a spustíme server:

```
$ cd hg1
$ hg serve                    # Použije se flag -d pro režim daemonize
```

Ve webovém prohlížeči otevřeme stránku http://localhost:8000 a dostaneme webové rozhraní pro zkoumání historie, tagů, větví, atd.

Tutéž URL adresu můžeme použít z příkazového řádku:

```
$ cd ..
$ hg clone http://localhost:8000 my_cloned_repo
```

9.5 Sledování externího softwaru

Někdy jsou používány komponenty třetí strany a ty mohou požadovat jistou uživatelskou úpravu, aby vyhovovaly lokálním potřebám. Jak sledovat externí software a přitom zachovat lokální uživatelské nastavení? Chceme:

1. importovat nové verze kódů
2. zachovat snadnou použitelnost našich změn v nově importovaných verzích.

S malým úsilím to lze s použitím Mercurialu provést. Buďto použijeme dva repozitáře, nebo jen jeden s importovanou větví.

9.5.1 Použití dvou repozitářů

Kód *Expat,* který použijeme v příkladech, je parsovací knihovna XML.

Vytvoříme adresář a repozitář, importujeme externí verzi, vytvoříme klon a upravíme jej:

```
$ mkdir expat-import
$ wget http://sourceforge.net/projects/expat/files/expat/1.95.0/expat-
1.95.0.tar.gz/download
$ tar xvfz expat-1.95.0.tar.gz
...
$ cp -ar expat-1.95.0/* expat-import
$ cd expat-import
$ hg init
$ hg addremove
...
$ hg ci -m"First check-in"
$ hg tag 1.95.0
$ cd ..
$ hg clone expat-import myexpat
updating working directory
56 files updated, 0 files merged, 0 files removed, 0 files unresolved
$ cd myexpat
$ gedit lib/expat.h
```

Soubor lehce změníme (přidáme příkaz #ifdef , jak vidíme v diffu dole), uložíme a komitujeme:

```
$ hg diff
diff -r ed48f733d885 lib/expat.h
--- a/lib/expat.h       Fri Jan 01 18:54:59 2010 +0100
+++ b/lib/expat.h       Fri Jan 01 19:00:54 2010 +0100
@@ -8,6 +8,10 @@

 #include <stdlib.h>

+#ifdef MY_PATCH
+int MyPatchVar;
+#endif
+
 #ifndef XMLPARSEAPI
 #  ifdef __declspec
 #    define XMLPARSEAPI __declspec(dllimport)
$ hg ci -m"My patch"
```

Stáhneme další verzi a aktualizujeme pracovní adresář *expat-import* :

```
$ cd ..
$ wget http://sourceforge.net/projects/expat/files/expat/1.95.1/expat-
1.95.1.tar.gz/download
$ tar xvfz expat-1.95.1.tar.gz
...
$ cp -ar expat-1.95.1/* expat-import
$ cd expat-import
$ hg ci -m"Imported 1.95.1"
$ hg tag 1.95.1
```

Přetáhneme změny a přeskupíme:

```
$ cd ../myexpat
$ hg pull --rebase                    # Use rebase to keep your patch on top
pulling from /home/thorman/expat-import
searching for changes
adding changesets
adding manifests
adding file changes
added 2 changesets with 14 changes to 14 files (+1 heads)
(run 'hg heads' to see heads, 'hg merge' to merge)
merging lib/expat.h
saving bundle to /home/thorman/myexpat/.hg/strip-backup/9451a841b310-temp
adding branch
adding changesets
adding manifests
adding file changes
added 3 changesets with 15 changes to 14 files
rebase completed
$ hg glog
@  changeset:   3:653e4db40e6b
|  tag:         tip
|  user:        Thorbjorn Jemander <thorbjorn@jemander.se>
|  date:        Fri Jan 01 19:01:52 2010 +0100
|  summary:     My patch
|
o  changeset:   2:36287c5314a4
|  user:        Thorbjorn Jemander <thorbjorn@jemander.se>
|  date:        Fri Jan 01 19:05:59 2010 +0100
|  summary:     Added tag 1.95.1 for changeset 15d033d27d73
|
o  changeset:   1:15d033d27d73
|  tag:         1.95.1
|  user:        Thorbjorn Jemander <thorbjorn@jemander.se>
|  date:        Fri Jan 01 19:05:50 2010 +0100
|  summary:     Imported 1.95.1
|
o  changeset:   0:ed48f733d885
   tag:         1.95.0
   user:        Thorbjorn Jemander <thorbjorn@jemander.se>
   date:        Fri Jan 01 18:54:59 2010 +0100
   summary:     First check-in
```

Přeskupování není nezbytné, ale usnadňuje to pozdější změnu oprávky: jednoduše editujeme tip a

provedeme komit.

9.5.2 Použití pojmenované větve

Založíme repozitář a vytvoříme větev s názvem import; importovaný obsah přidáme do této větve.

```
$ mkdir expat; cd expat; hg init; cd ..
$ cp -ar expat-1.95.0/* expat
$ cd expat
$ hg addremove
...
$ hg ci -m"First check-in"
$ hg branch import
marked working directory as branch import
$ hg tag 1.95.0
$ hg update default
0 files updated, 0 files merged, 1 files removed, 0 files unresolved
$ gedit lib/expat.h
```

Změníme soubor, zaznamenáme jej, přejdeme do větve *import*, importujeme další verzi a sloučíme:

```
$ hg ci -m"My patch"
created new head
$ hg up import                      # Move to import branch
2 files updated, 0 files merged, 0 files removed, 0 files unresolved
$ cp -ar ../expat-1.95.1/* .
$ hg ci -m"Imported 1.95.1"
$ hg tag 1.95.1
$ hg up default                     # Move to default branch
13 files updated, 0 files merged, 1 files removed, 0 files unresolved
$ hg merge -r 4
merging lib/expat.h
13 files updated, 1 files merged, 0 files removed, 0 files unresolved
(branch merge, don't forget to commit)
$ hg glog
@    changeset:   5:a60f4583b4f3
|\   tag:         tip
| |  parent:      2:b75f6b9f470f
| |  parent:      4:1e367bc40b34
| |  user:        Thorbjorn Jemander <thorbjorn@jemander.se>
| |  date:        Fri Jan 01 18:39:37 2010 +0100
| |  summary:     Merged in new import
| |
| o  changeset:   4:1e367bc40b34
| |  branch:      import
| |  user:        Thorbjorn Jemander <thorbjorn@jemander.se>
| |  date:        Fri Jan 01 18:37:31 2010 +0100
| |  summary:     Added tag 1.95.1 for changeset 26c2874fe905
| |
| o  changeset:   3:26c2874fe905
| |  branch:      import
| |  tag:         1.95.1
| |  parent:      1:d126eef1462c
| |  user:        Thorbjorn Jemander <thorbjorn@jemander.se>
| |  date:        Fri Jan 01 18:37:21 2010 +0100
| |  summary:     Imported 1.95.1
| |
```

```
o |  changeset:   2:b75f6b9f470f
|/   user:        Thorbjorn Jemander <thorbjorn@jemander.se>
|    date:        Fri Jan 01 18:35:46 2010 +0100
|    summary:     Implemented my patch
|
o  changeset:   1:d126eef1462c
|  branch:      import
|  user:        Thorbjorn Jemander <thorbjorn@jemander.se>
|  date:        Fri Jan 01 18:32:20 2010 +0100
|  summary:     Added tag 1.95.0 for changeset ade3e08739f3
|
o  changeset:   0:ade3e08739f3
   branch:      import
   tag:         1.95.0
   user:        Thorbjorn Jemander <thorbjorn@jemander.se>
   date:        Fri Jan 01 18:32:11 2010 +0100
   summary:     First import
```

Osobně dávám přednost metodě se dvěma repozitáři ve spojení s přeskupením.