Práce se systémem Subversion (SVN)

© **2007-2008 Josef Pelikán, CGG MFF UK Praha**

http://cgg.ms.mff.cuni.cz/~pepca/svn/

Josef.Pelikan@mff.cuni.cz

Obsah

- VCS systémy obecně
- Subversion – základní koncepce
- pracovní cyklus, nejčastěji používané příkazy
- organizace projektu a práce s repository
- pokročilejší postupy:
  - příkaz copy, tagging, diff
  - branching, merging, backporting
- klient TortoiseSVN

Version Control Systems

- **správa projektů**, správa verzí („verzování“)
  - ukládání historie celého projektu
  - snadné zálohování
  - spolupráce více vývojářů na společné množině souborů
  - správa zveřejněných verzí, podpora zpětné portace chyb
  - různé statistiky pro šéfa projektu, vedení firmy :-)
- asi nejznámějším systémem je **CVS**
  - další (i komerční): ClearCase, Visual SourceSafe, GNU Arch, Bazaar, Perforce, BitKeeper, Microsoft Team Foundation, SCCS, RCS

Přístupy

- uložení dat: **distribuované** vs. **centralizované** (klient-server)
- paralelní modifikace: **„merge“** vs. **„lock“**
  - „merge“ - je dovoleno paralelně editovat stejný soubor, konflikty se řeší až při commitu
  - „lock“ - vývojář soubory před editací zamyká
- historie: **„changesets“** vs. **„snapshots“**
- **atomický commit?** (konzistence repository)
- meta-data? verzování adresářů? přejmenování / klonování souborů? mazání? sledování větví / slévání?

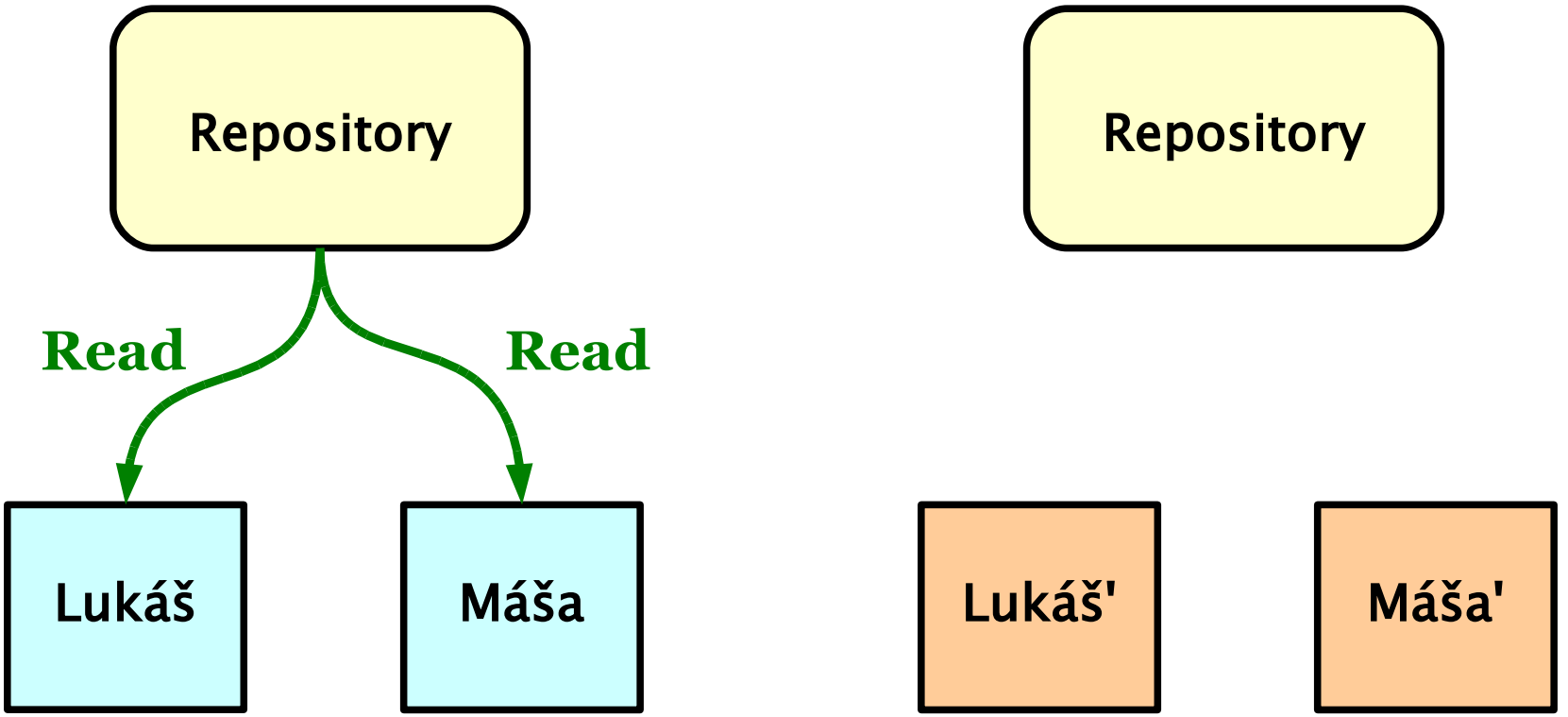
Subversion (SVN) – koncepce I

- **klient-server**
  - repository je primárně uložená na serveru
- globální **„časová osa“**
- **atomické commity** (konzistence repository)
- **copy – modify – merge**
  - lokální kopie repository (její části): **checkout** / **update**
  - **modifikace** lokální kopie (off-line, udržuje se „base“)
  - **update** + **merge**: řeší se případně konflikty vzniklé paralelní editací
  - **commit**: odeslání změn do repository na server

Subversion – koncepce II

- **lock – commit**
  - doporučuje se v případě binárních souborů, kde nejde rozumně dělat kontextové slévání („merge“)
- **meta-data**, **adresáře** – jsou verzovány
- **diference** se komprimují a posílají oběma směry
  - to se týká i binárních souborů
- **kopie/klony** (.. na straně serveru) jsou **velmi levné**!
- **omezení:**
  - mazání souboru + následné použití stejného jména
  - není automaticky udržován pořádek ve větvení a slévání

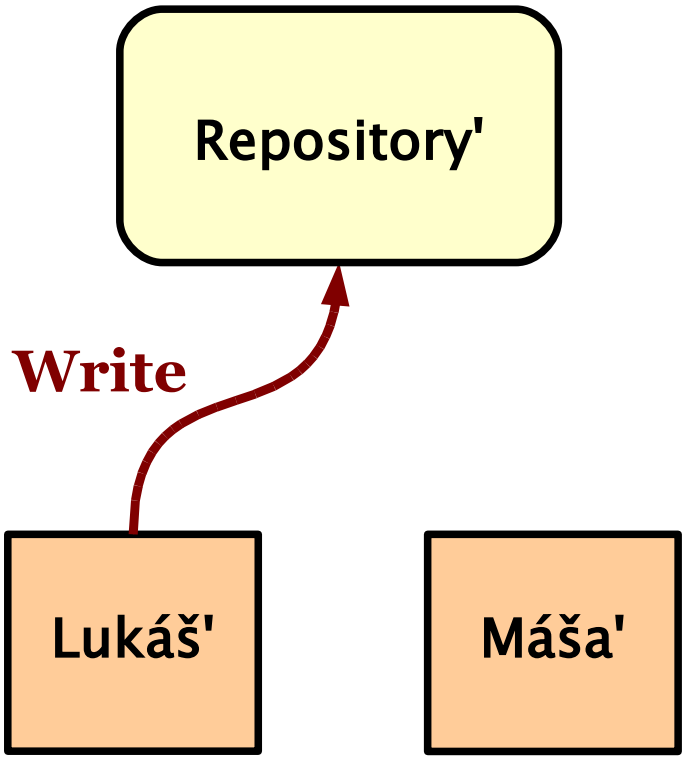
Paralelní práce na projektu



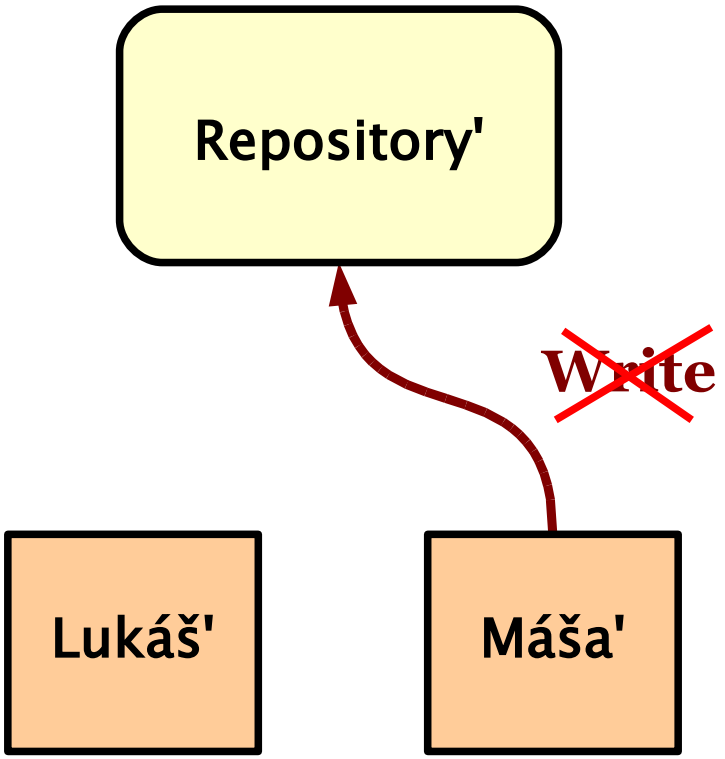
Oba mají totožná data

Oba editují svá data

Vznik konfliktu

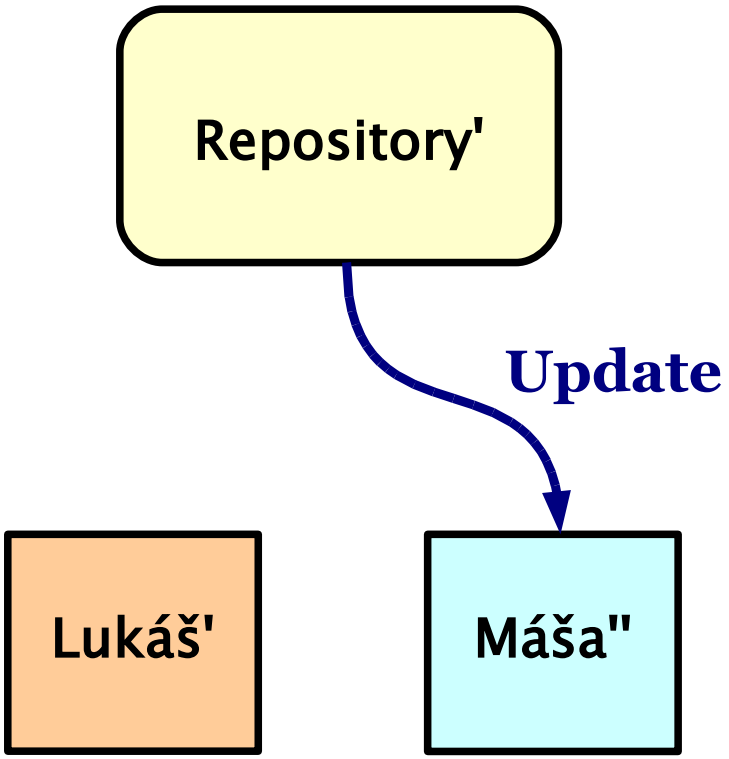


Lukáš posílá svoje opravy

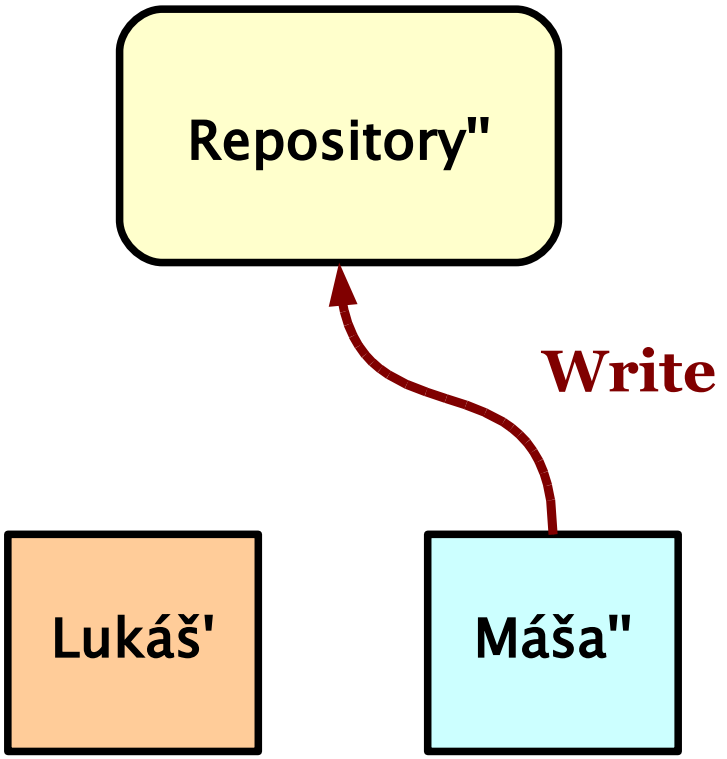


Máša by chtěla poslat opravy
⇒ konflikt („out of date“)

Schéma „copy – modify – merge“

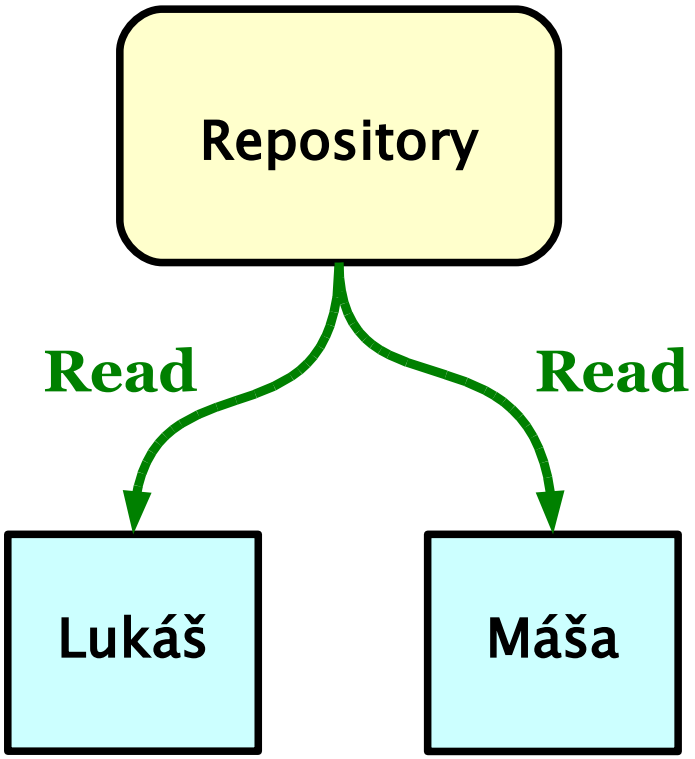


Máša přijímá Lukášovy opravy (příp. „merge“)

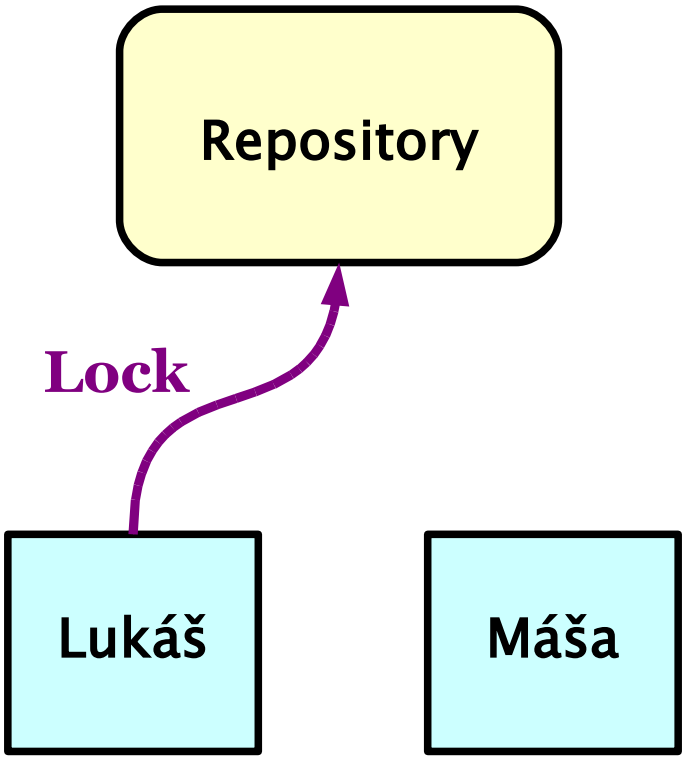


Máša může poslat své opravy (jsou v souladu s Lukášovými)

Zamykání (exkluzivní editace)

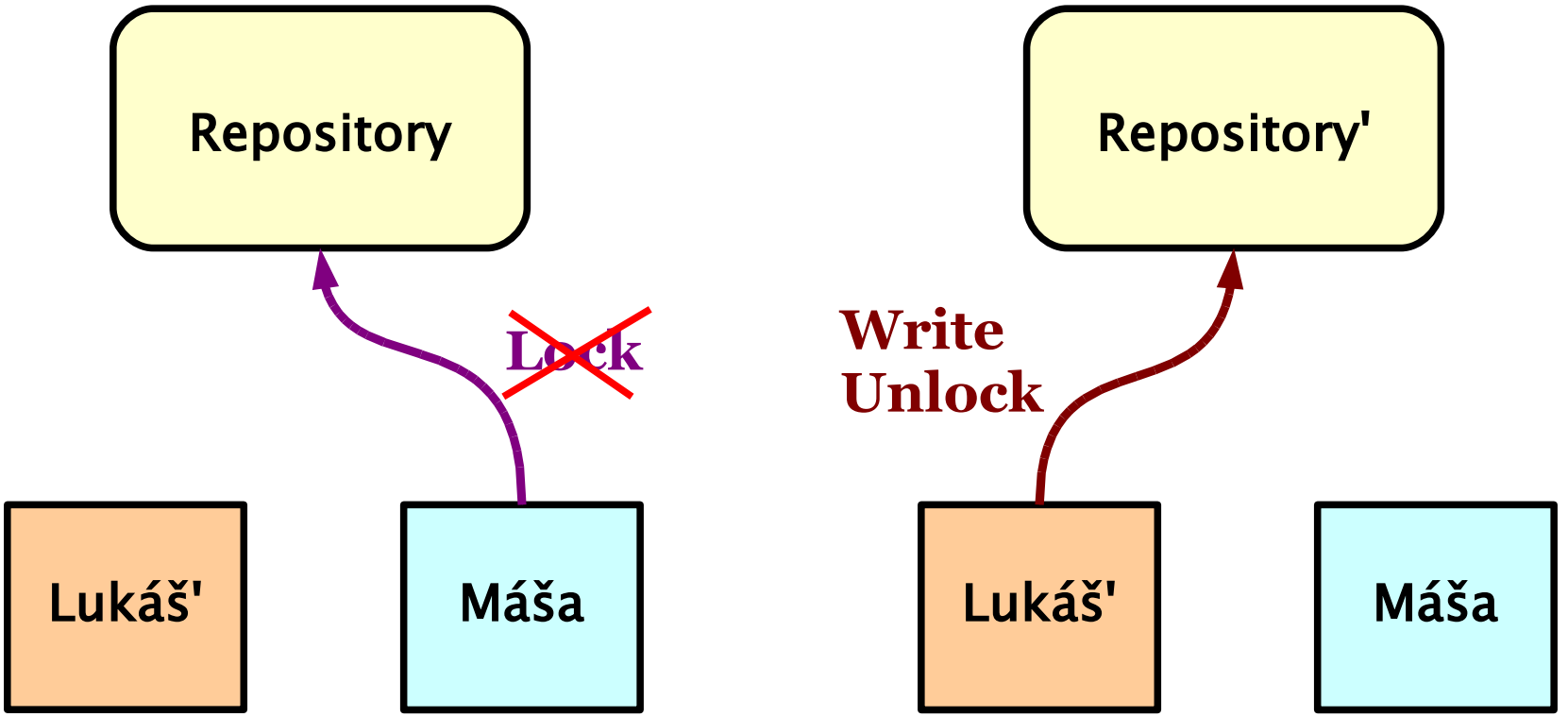


Oba mají totožná data



Lukáš chce editovat,
zamkne si příslušné soubory.

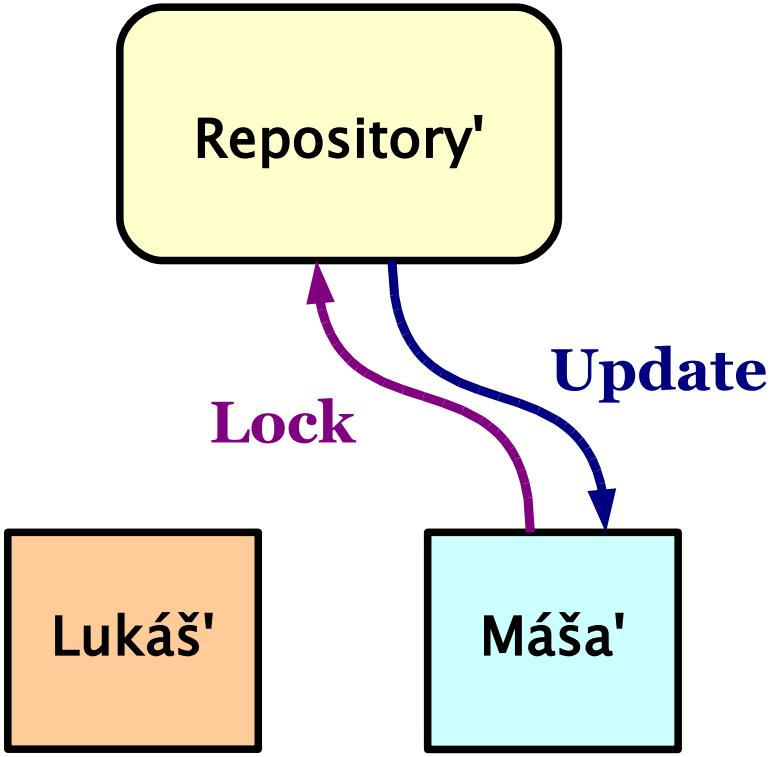
Zamykání (exkluzivní editace)



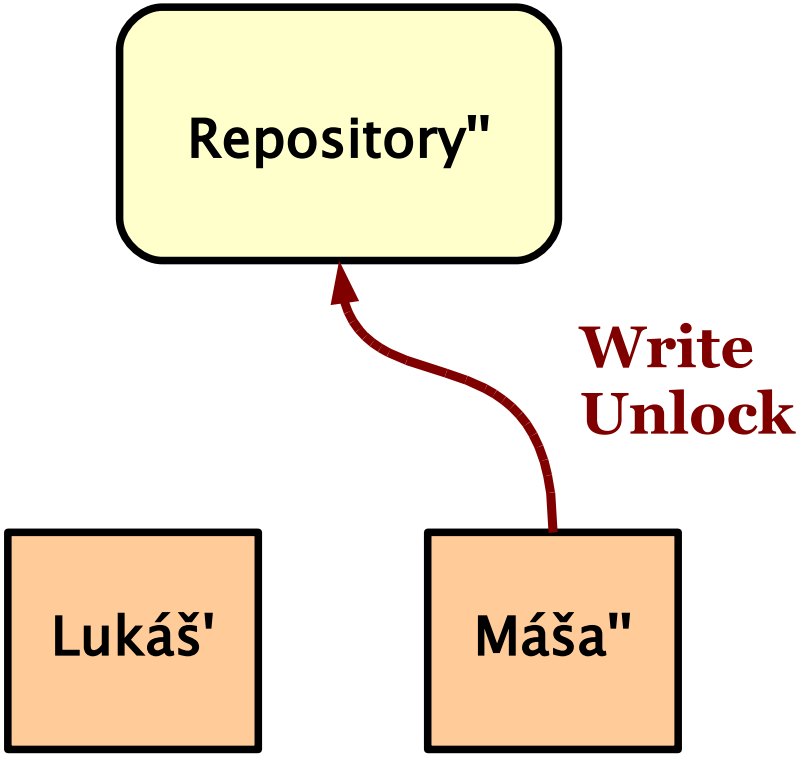
Máša chce také editovat, ale zámek nedostane!

Lukáš skončil editaci, uvolňuje zámek.

Zamykání (exkluzivní editace)



Máša už teď zámek dostane.



Po editaci a odeslání změn i Máša uvolňuje zámek.

Pracovní kopie („working copy“)

- **kopie repository nebo její části** na pracovní stanici
  - každý vývojář má vlastní pracovní kopii (např. na lokálním disku)
- počáteční download – „**checkout**“
- aktualizace pracovní kopie z repository – „**update**“
  - umí **zachovat lokální úpravy**, pokud nejsou ve velkém rozporu se změnami z repository (tj. udělá jakési „slévání“ změn)
  - pokud jsou lokální a veřejné změny v konfliktu, je lokální vývojář požádán o jejich vyřešení („resolving“), to se musí dělat ručně

Běžný pracovní cyklus

- aktualizace pracovní kopie: „**checkout**“ nebo „**update**“
- **editace** pracovní kopie
  - neměla by trvat příliš dlouho, pokud pracuji v živém týmu
- aplikace změn z repository: „**update**“
  - jde hlavně o změny, které mezitím udělali vývojáři na stejných souborech
  - ale i jiné úpravy mohou např. pokazit kompilaci, apod.
  - případné ruční **rešení konfliktů**, nakonec: „**resolved**“
- odeslání lokálních změn do repository: „**commit**“

Editace pracovní kopie

Běžná editace již **existujících souborů** vývojovými nástroji a navíc:

- **přidání** nového souboru/adresáře: „**add**“
- **odstranění** z repository: „**delete**“
- **kopie** souboru/adresáře (klon, zachová se historie): „**copy**“
- **přejmenování**/přesun do jiného adresáře: „**move**“
- **vrácení** lokálně provedených změn: „**revert**“
  - změny od posledního příkazu „commit“/“update“ se zruší

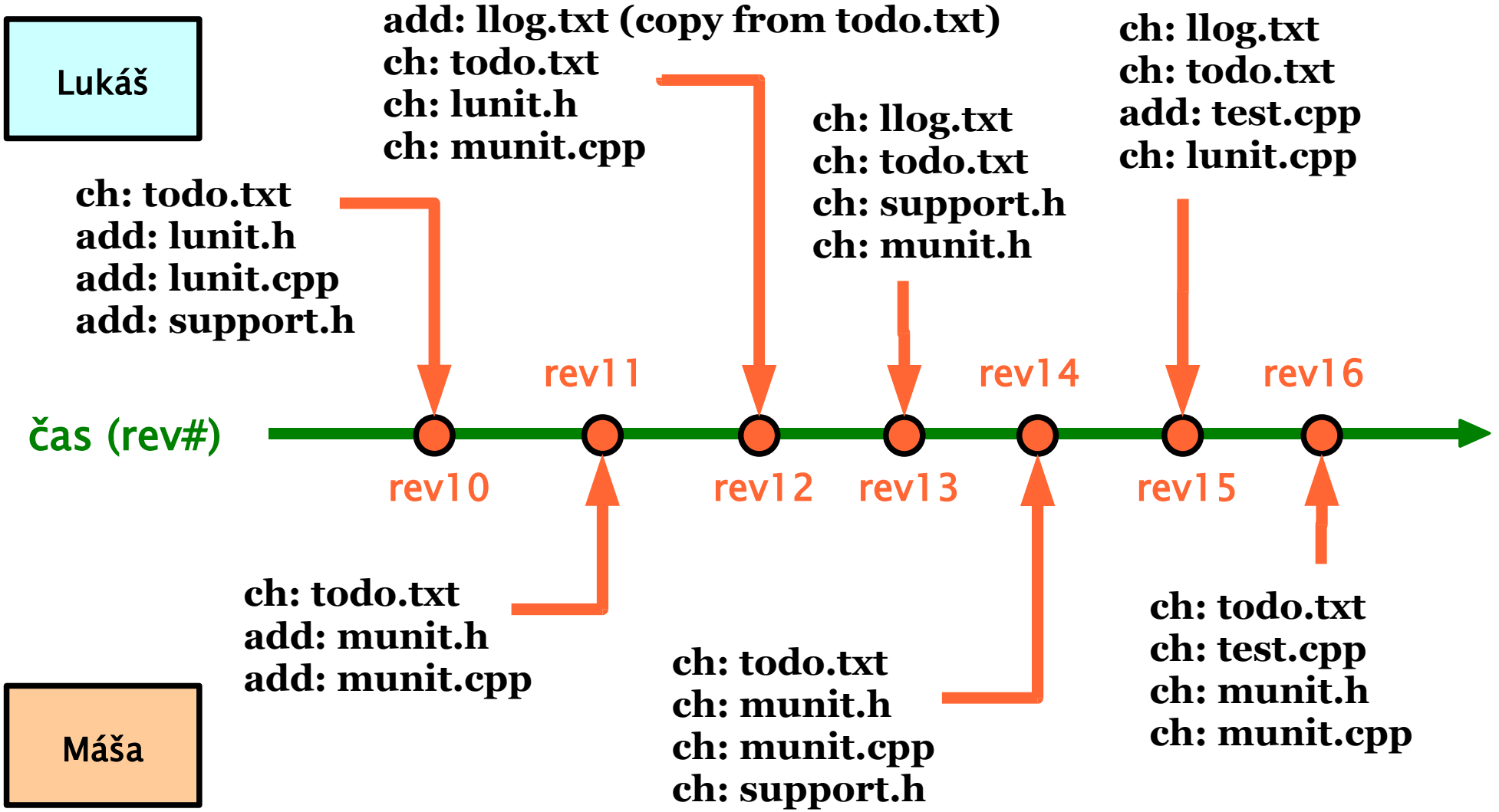
Změny v pracovní kopii

- systém SVN si u celé vaší kopie pamatuje, z jakého stavu jste při editaci vycházeli (umí pracovat „off-line“!)
- je možné se **vrátit k původnímu stavu** („pristine copy“) příkazem: „**revert**“
  - potenciálně nebezpečné, můžete přijít o svou práci!
- k prohlížení změn slouží příkaz: „**status**“
- rozdíly konkrétního souboru: „**diff**“
  - má i obecnější použití (změny mezi různými verzemi/revizemi libovolných souborů)

Model historie v SVN

- koncepce „**snapshots**“:
  - z hlediska serveru jsou jedinými okamžiky, kdy se repository mění, úspěšně dokončené operace „**commit**“ (viz „atomický commit“)
  - jednotlivé commity jsou na časové ose uspořádané – očíslovaným stavům repository se říká „**revize**“
- **čísla revizí** jsou společná pro celou repository
  - snadno se vrátím v čase přesně do daného okamžiku – a to jak pro konkrétní soubor, tak i pro množinu souborů nebo adresářů
  - stav pracovní kopie však může být složitější (mix revizí)

Příklad vývoje repository



Příklad: verze jednotlivých souborů

	rev 9	rev 10	rev 11	rev 12	rev 13	rev 14	rev 15	rev 16
todo.txt	?	10	11	12	13	14	15	16
support.h		10	10	10	13	14	14	14
lunit.h		10	10	12	12	12	12	12
lunit.cpp		10	10	10	10	10	15	15
munit.h			11	11	13	14	14	16
munit.cpp			11	12	12	14	14	16
llog.txt				12	13	13	15	15
test.cpp							15	16

červená změna souboru v dané revizi
modrá klon existujícího souboru („svn copy“)

Zkoumání historie

- je třeba pracovat „**on-line**“, pracovní kopie historii neobsahuje
- výpis **seznamu změn** („commitů“) konkrétního souboru nebo adresáře: „**log**“
  - jednotlivé položky obsahují: číslo revize, datum a čas, autor, **poznámka ke commitu**
  - pište smysluplné **poznámky ke commitům**
  - SVN nedovolí poznámku úplně vynechat (ale ...)
- přehled posledních změn textového souboru po jednotlivých řádcích: „**blame**“

Zkoumání historie: diference

- výpis nebo zobrazení **rozdílů** mezi verzemi jednotlivých souborů/adresářů: „**diff**“
  - nejjednodušší verze příkazu ukáže změny od „updatované“ verze („pristine copy“)
  - lze generovat změny mezi libovolnými verzemi a/nebo na několika souborech najednou
  - lze generovat změny ve formátu „**patch**“
  - efektivní pro textové soubory s rozumně krátkými řádky
- pohled dovnitř SVN:
  - interně se diference počítají i u binárních souborů a do databáze repository se ukládají právě jen tyto diference

Subversion server

- SVN repository může být přístupná pomocí různých protokolů
  - **file:///** – data jsou přímo uložna ve FS klientského počítače (problémy s paralelním přístupem, bezpečnost)
  - **http://** – WebDAV přístup přes server Apache
  - **https://** – jako http, navíc šifrování pomocí SSL
  - **svn://** – speciální protokol (port 3690), uživatelské účty, „authentication realms“
  - **svn+ssh://** – jako svn, ale skrz SSH tunel

Meta-data, atributy

- soubory/adresáře pod kontrolou SVN mohou být opatřeny různými „**atributy**“
  - „content-type“ souboru, ošetření konců řádků, ignorování souborů v adresářích, … + vlastní uživatelské atributy
  - atributy se „**verzují**“ jako všechna ostatní data
- je možné nakonfigurovat **automatické nastavování atributů**
  - např. každý soubor „***.txt**“ je označen jako textový s nativními konci řádků
  - nebo každý soubor „***.sh**“ je označen jako textový a navíc má atribut „**executable**“, ..

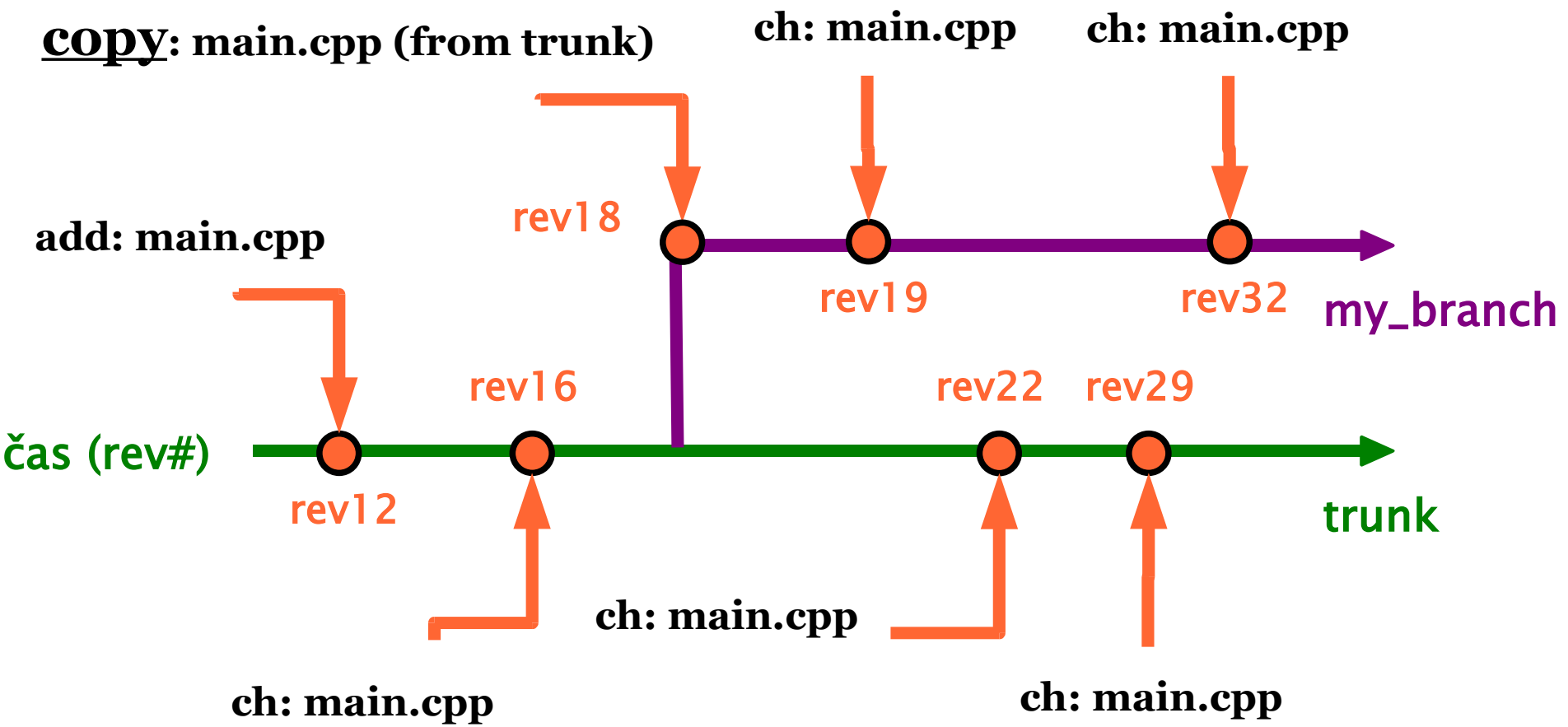
Populární vestavěné atributy

- **svn:eol-style** – „native“, „CRLF“, „CR“, „LF“
  - ošetření konců řádků (přenos mezi klientem a serverem)
- **svn:mime-type** – klasický MIME-type
- **svn:executable** – pro checkout do UNIXu
- **svn:ignored** – seznam položek, které se ignorují
- **svn:keywords** – substituce v textových souborech
  - „**Date**“ – zapíná substituci „$Date: $“ (datum commitu)
  - „**Rev**“ – zapíná substituci „$Rev: $“ (číslo revize)
  - „**Author**“ – substituce „$Author: $“ (kdo commitoval)
  - „**Id**“ – „$Id: $“ se nahradí zkratkou předchozích info.

Větvení, slévání, .. („branching“)

- udržování několika málo se lišících **verzí projektu**
  - s možností snedného přenášení oprav z jedné větve do druhé (něco jako „backporting“ .. „crossporting“)
- **refaktorizace**, přestavba v nezávislé větvi
  - práce bude trvat delší dobu
  - nechceme, aby se porušila kompilovatelnost/funkčnost hlavní větve
- podobné techniky se používají i k vytváření „**snapshots**“ nebo „**tags**“
  - ale nepředpokládá se, že se „tag“ bude někdy měnit

Příklad rozvětvení (kvůli refaktorizaci)



Operace „copy“

- základem pro vytváření **větví**, „**tags**“, apod.
  - interně se implementuje jen pomocí **odkazů**
  - je velice **levná**! („copy-on-change“)
- představa: v jistém okamžiku se objekty (soubory, adresáře) **naklonují**
  - převezmou **kompletní historii** od svých vzorů
  - přebírají se samozřejmě i **atributy** (a jejich historie..)
- **cílové umístění** kopie (klonu) je libovolné
  - musí být ve **stejné repository**
  - konvence: adresáře „**branches**“, „**tags**“, „**trunk**“

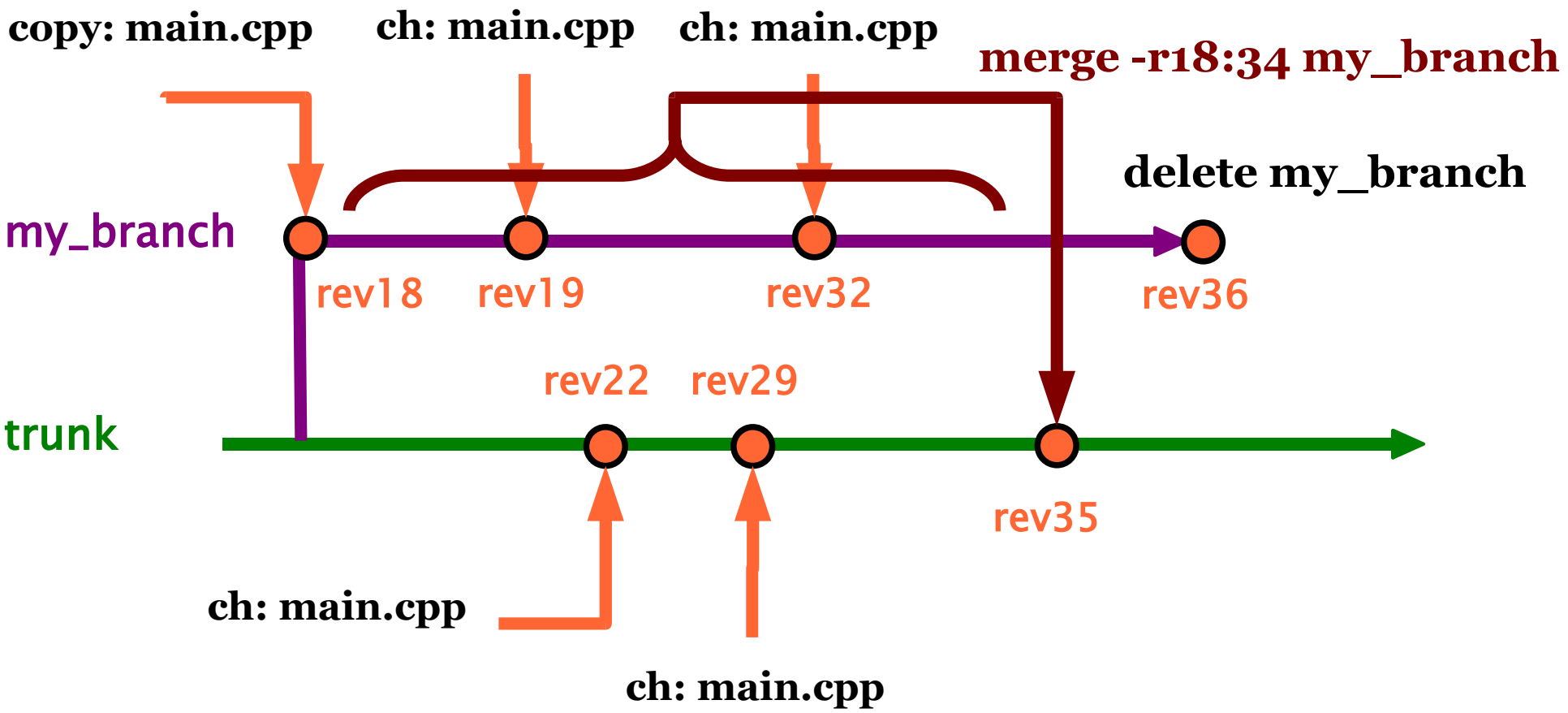
Branch a pracovní kopie

- po „commitování“ operace „**copy**“ se ve file-systému SVN repository vytvoří klon daného adresáře
  - vývojář může udělat „**checkout**“ nebo „**update**“ a začít pracovat v tomto novém adresáři
  - nebo je možné v pracovní kopii nastavit, že má být dosavadní adresář „**přepnut**“ do nové větve:
- příkaz „**switch**“
  - korektní postup, jak v pracovní kopii „přepsat“ objekt (adresář/soubor) objektem z jiné větve
  - je podobný příkazu „update“ s odlišným URL
  - přehnané používání příkazu „switch“ ⇒ **zmatek!**

Přenášení úprav mezi větvemi

- příkaz „**merge**“:
  - přenese **specifikované změny** (úpravy, posloupnosti úprav) z jedné větve do druhé
  - korektně ošetřuje všechny „editační“ operace včetně změn atributů
  - **historie přenášení** (které změny byly již přenesesny a které ne) se zatím v SVN musí dělat ručně
  - pozor, aby nebyla některá změna přenesena **vícekrát**!
- „**svn merge –dry-run ..**“
  - provede operaci „nanečisto“ – aspoň tak se dá konfliktům zabránit

Příklad „slévání větví“



Přenášení úprav mezi větvemi

- „**merge**“ z trunku **do vedlejší větve** (synchronizace)
  - měl by se provádět dost často, jinak je potom přenos zatížen mnoha konflikty
- „**merge**“ **z vedlejší větve** do trunku
  - pokaždé, když potřebujeme odrazit [mezi-]výsledek v hlavní větvi
  - testeři mohou naše opravy začít testovat
- finální přenesení změn a **zánik vedlejší větve**
  - smazání již nepotřebné větve – příkaz „**delete**“

Další použití „copy“ a „merge“

- „**copy**“ slouží i k vytváření **zálohy** („snapshot“, „**tag**“) aktuálního stavu hlavní větve
  - organizační konvence – v takové záloze se již nesmí dělat úpravy!
  - příklad: adresář „**tags**“ – sbírka všech „public releases“
- selektivní „**merge**“ (z trunku) do některé zálohy
  - záloha by měla být naklonovaná – viz výše
  - „**backporting**“ **oprav chyb** do starších verzí – musí se dát pozor na aplikovatelnost změn
  - případné nesrovnalosti se musí řešit ručně

Co se nevešlo..

- „**export**“: export stromu z repository na disk
  - neobsahuje řídící adresáře, nevznikne pracovní kopie
- „**import**“: import dat do repos. a okamžitý „commit“
- „**cleanup**“: regenerace řídících adresářů
- „**lock**“ a „**unlock**“: zamykání (exkluzivní editace)
- „**svnadmin create**“: na serveru – vytvoří novou repo.
- „**svnadmin dump**“ resp. „**svnadmin load**“
  - zálohování do textového souboru resp. obnova repository ze zálohy

Organizace repository

- adresář „**trunk**“:
  - poslední funkční verze, veřejně přístupná?
  - měla by jít **přeložit** a **fungovat** (? výjimky ?)
- adresář „**branches**“:
  - když se nějaký vývojář chce déle „vrtat“ v kódu
  - často synchronizaovat s trunkem!
- adresář „**tags**“:
  - seznam release verzí, „milestones“
  - nesmí se editovat! (technicky lze zajistit pomocí „hook“)
- zvláštní případy: větve pro různé zákazníky, HW, ...

Co do repository patří

- co je předmětem vývoje, co se (nejčastěji ručně) edituje, co chceme verzovat
  - **zdrojáky**
  - **projektové soubory**, „**Makefiles**“
  - Microsoft Visual Studio: *.dsw, *.dsp, *.sln, *.vcproj
  - **dokumentace**, texty, návody (prefer. ve zdrojáku)
  - „log-files“, „todo-files“
  - **nejnutnější data** patřící k projektu (ikony, konfigurační soubory)
  - soubory, které nejsou moc dlouhé a je obtížné je stáhnout jinde..

Co do repository nepatří!

- obecně: co lze zkompilovat, transformovat ze souborů z repository, co se u každého vývojáře liší..
  - **výsledky překladu** (*.exe, *.class, *.jar, *.so, apod.)
  - **pracovní a dočasné soubory** (*.obj, *.o, *~, *.bak, apod.)
  - Microsoft Visual Studio: *.ncb, *.opt, *.plg, *.suo, *.user
  - **komponenty** (knihovny) **třetích stran**, které jsou snadno dostupné jinde (? výjimky ?)

Organizace práce I

- **„commitovat“ co nejčastěji**
  - minimálně jednou denně
  - vždy po uzavření logického celku (oprava **jedné** chyby..)
- psát **poznámky** ke „commitům“
  - stručné a informativní
  - nepsat, co se změnilo (to každý vidí), ale **jak** a **proč**!
  - oprava chyby („kartička“) ⇒ **číslo případu**
- **vedlejší větve** („branches“)
  - potřeba dlouho pracovat a nerozbít projekt ostatním
  - nenechat větev příliš zastarat (synchroniz. oběma směry)

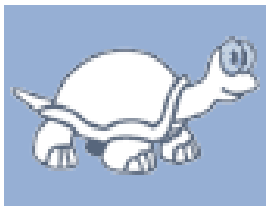
Organizace práce II

- **vlastnictví kódu** vs. **„bazar“**
  - každý programátor zodpovídá za jisté moduly
  - dovolím opravovat cokoli komukoli, ale vlastník to nakonec zkontroluje (automatické hlášení přes „hooks“)
- **trunk** se udržuje funkční (minimálně přeložitelný!)
  - polední kompletní build (viz Spolsky)
  - hříšník builduje příště..

- využívat **SVN-atributy**
  - „svn:ignored“, „svn:eol-style“, „svn:keywords“

TortoiseSVN

- **grafický SVN klient** integrovaný do **Windows**
  - *shell extension* ⇒ lokální menu v každém file-exploreru
  - *repository browser* (registrace URI protokolů „svn://“ ..)
  - některé příkazy na *pravé tlačítko myši*! („copy“, „move“)
- velmi pohodlné **GUI verze** příkazů:
  - **„diff“, „merge“**
  - **„log“, „blame“, „revision graph“**



TortoiseSVN

Literatura, on-line zdroje

- **SVN home-page**: http://subversion.tigris.org/
- **Subversion-book**: http://svnbook.red-bean.com/
- klient **TortoiseSVN** pro Windows: http://tortoisesvn.tigris.org/
- multi-platformní klient **RapidSVN**: http://rapidsvn.tigris.org/
- **Joel on Software**: http://joelonsoftware.com/