[MAC0211] Laboratório de Programação I
Aula 23
Sistemas de Controle de Versão

Kelly Rosa Braghetto

DCC–IME–USP

06 de junho de 2013

Na aula passada...

- Geradores de Analisadores Léxicos – Flex (Parte 2)
- Geradores de Analisadores Sintáticos – Bison

Controle de versão (da Engenharia de Software)

- Controle de versão = versionamento = controle de revisão
- Se refere ao gerenciamento de mudanças em documentos, códigos fontes de programas, páginas web, e outras coleções de informação
- Geralmente, mudanças são identificadas por um código numérico ou alfa-numérico, chamado de *revisão* ou *número da revisão*
- É uma tarefa complexa, especialmente quando temos grupos de pessoas alterando os mesmos arquivos

Sistema de Controle de Versão

É um sistema com os seguintes propósitos:

- **permitir que múltiplas pessoas de uma equipe trabalhem simultaneamente em um mesmo projeto**
  $\Rightarrow$ cada pessoa altera suas cópias dos arquivos e escolhe quando quer compartilhar as mudanças com o resto da equipe. Assim, alterações parciais ou temporárias de uma pessoa não interferem no trabalho de outras.
- **permitir que uma pessoa use múltiplos computadores para trabalhar em um projeto**
  $\Rightarrow$ O sistema é útil mesmo quando a pessoa está trabalhando sozinha.

Sistema de Controle de Versão (cont.)

É um sistema com os seguintes propósitos:

- **integrar o trabalho feito simultaneamente por diferentes membros de uma equipe**

  $\Rightarrow$ Na maior parte do tempo, alterações em diferente arquivos ou até mesmo num mesmo arquivo podem ser combinadas sem que nenhum trabalho feito seja perdido. Mas quando duas ou mais pessoas fazem alterações conflitantes em uma mesma linha de um arquivo, então o sistema de controle de versão solicita a ajuda humana para decidir como lidar com o conflito.
- **manter o histórico de versões de um projeto**

  $\Rightarrow$ Isso é uma garantia contra perda de dados. Se uma pessoa comete um erro em uma alteração, ela pode recuperar uma versão anterior do arquivo. É possível reproduzir ou entender um erro reportado em um versão antiga de um software, ou então determinar quando, porque e por quem qualquer parte de um arquivo foi alterada.

Repositórios e cópias locais

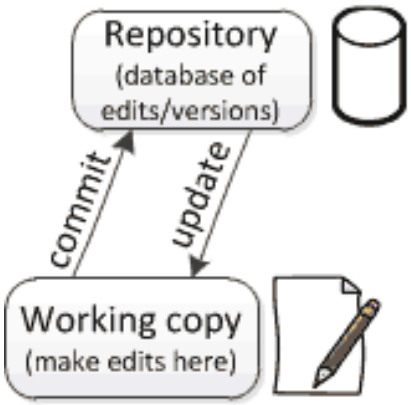
Um sistema de controle de versão básico usa:

- Um **repositório**, que é um banco de dados de todas as alterações e/ou versões históricas (*snapshots*) dos arquivos de um projeto
- **Cópias locais** (*working copies* = *checkouts*), que são cópias pessoais dos arquivos do projeto. As alterações feitas nessas cópias não afetam os demais membros da equipe do projeto

$\Rightarrow$ Um servidor pode ter vários sistemas de controle de versão
$\Rightarrow$ Um sistema de controle de versão pode ter vários repositórios

Repositórios e cópias locais

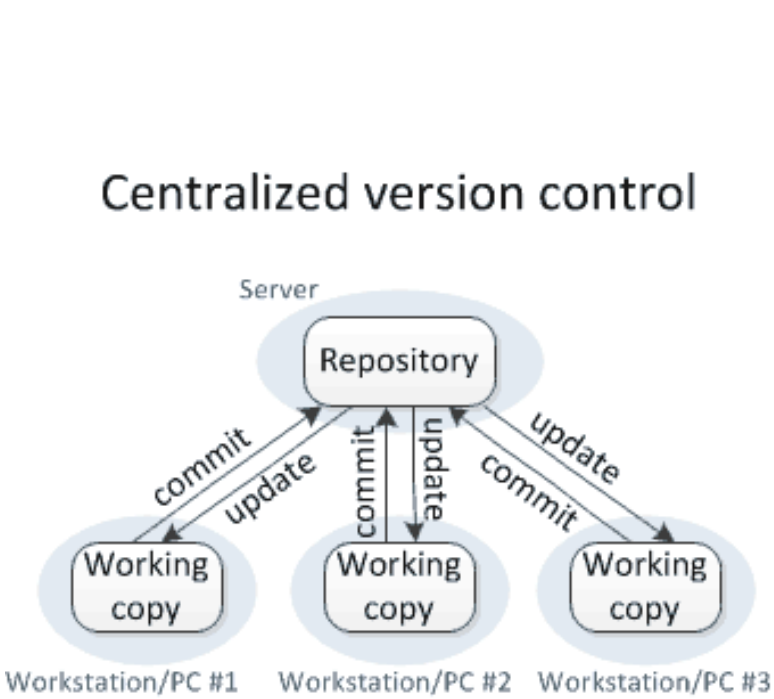
- As alterações feitas em cópias locais só são gravadas no repositório quando o responsável por elas faz a **submissão** (*commit*) dos arquivos alterados
- Um repositório pode conter alterações que não foram ainda aplicadas em uma dada cópia local. É preciso **atualizar** (*update*) uma cópia local para que ela incorpore todas as novas edições ou versões que foram adicionadas ao repositório desde a última vez em que a cópia foi atualizada
- Cada submissão produz uma nova versão no repositório, que representa "uma fotografia" atual dos arquivos

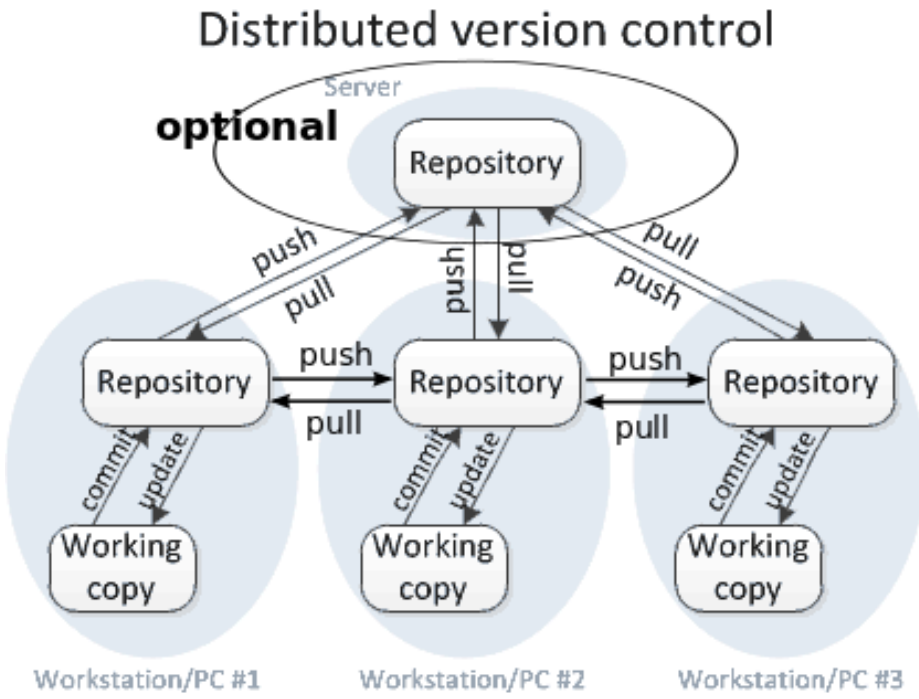


Controle de versão **centralizado** × **distribuído**

⇒ Principal diferença entre eles: número de repositórios

- Controle de versão centralizado – 1 só repositório por projeto
- Controle de versão distribuído – múltiplos repositórios por projeto





Figuras adaptadas de: `http://homes.cs.washington.edu/~mernst/advice/version-control.html`

Vantagens do controle de versão distribuído

- É possível fazer *commits* (gravar versões dos arquivos) sem disponibilizá-las para os demais membros do projeto
- É possível fazer *commits* mesmo quando o acesso a internet não está disponível
- Operações comuns (como *commits*, consulta ao histórico e reversão de alterações) são feitas de forma rápida, já que não requerem a comunicação com um servidor central
  $\Rightarrow$ A comunicação com os outros repositórios só é necessária quando se quer disponibilizar (*push*) ou obter (*pull*) as alterações para/de os outros membros do projeto
- Cada repositório local funciona como uma cópia do projeto, o que é uma proteção "natural" contra perda de dados

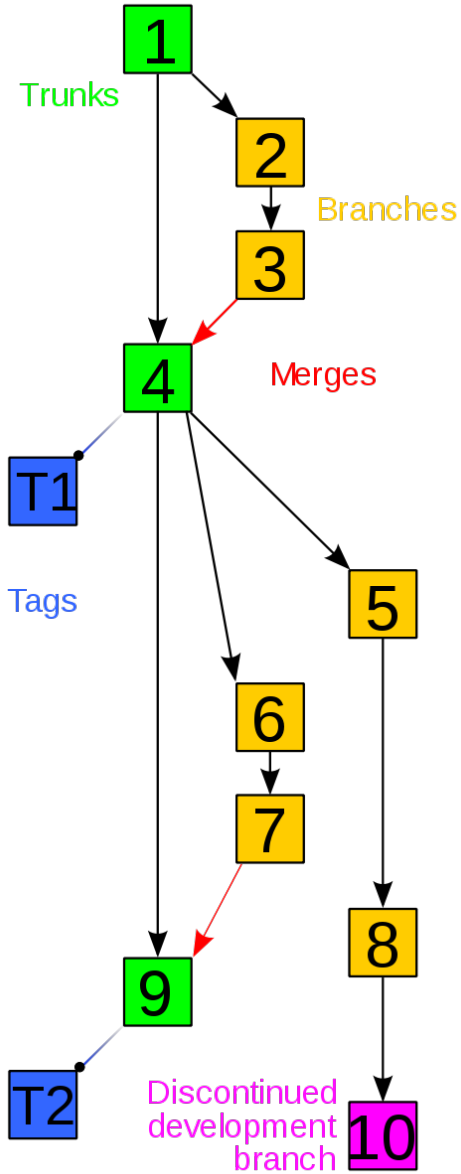
Alguns termos importantes

- **Raiz**, **linha principal** ou **braço principal** (*trunk*, *master*) – é o caminho de revisões que não se quebrou em um braço
- **Ramificação** ou **braço** (*branch*, *fork*) – quando a linha de desenvolvimento precisa ser dividida em duas ou mais
- Mesclagem inversa (*reverse integration*) – é quando um braço é mesclado à linha principal
- **Revisão** ou **versão** (*revision*, *version)* – representa um determinado "momento" (uma "fotografia") de um repositório ou documento
- **Marcação** (*tag*, *label release*) – é dar um nome a uma versão. Alguns sistemas centralizados não diferenciam entre "marcação" e "ramificação", pois é possível usar a ramificação com a finalidade de marcação

Alguns termos importantes ilustrados

Exemplo da visualização do histórico de um projeto usando um sistema de controle de versão:

Fonte da figura:
`http://en.wikipedia.org/wiki/Revision_control`



Sistemas de controle de versão centralizados

Exemplos famosos:

- **Revision Control System** (RCS) – `http://www.gnu.org/software/rcs/`
  Obs.: o primeiro sistema desse tipo de código livre.
  Usado apenas para arquivos locais, não tem servidor.
- **Concurrent Version System** (CVS) – `http://cvs.nongnu.org/` Obs.: o primeiro com servidor
- **Apache Subversion** (SVN) – `http://subversion.apache.org/`

Sistemas de controle de versão **distribuídos**

Exemplos famosos:

- **Git** – `http://git-scm.com/`
  Desenvolvido pelo Linus Torvalds; usado no desenvolvimento do Linux
- **Bazaar** (bzr) – `http://bazaar.canonical.com/en/`
  É um projeto GNU; usado no desenvolvimento do Ubuntu
- **Mercurial** (hg) – `http://mercurial.selenic.com/`
  Usado pelo Google, OpenOffice, Python, ...

Alguns sistemas vão além...

- **Fossil** – `http://www.fossil-scm.org/`
  Oferece mecanismos para: *bug tracker* distribuído, wiki distribuída e blog distribuído
- **Darcs** – `http://darcs.net/`
  Um controlador de versão *bastante* interativo e amigável. Tem outros diferenciais como, por exemplo, o suporte à integração de testes ao repositório (que permite definir um comando de teste – como o "make check" – que deve ser executado com sucesso para que alterações sejam gravadas de fato no repositório ou que atualizações possam ser feitas)
  Obs.: o Darcs não mantém um histórico cronológico das mudanças num repositório. Ele usa o conceito de *patches*.

Sites para repositórios *online*

Existem serviços de hospedagem web para projetos associados aos sistemas de controle de versão mais usados:

- **Github** (para Git) – `https://github.com/`
- **Launchpad** (para Bazaar) – `https://launchpad.net/`
- **Bitbucket** (para Mercurial) – `https://bitbucket.org/`

$\Rightarrow$ Esses sites funcionam como plataformas para o desenvolvimento colaborativo ("social") de software.

Boas práticas de programação

Com que frequência *commits* devem ser feitos?

- Para minimizar conflitos de versões e facilitar o controle histórico, recomenda-se que o código de um programa seja submetido sempre que ele estiver estável (ou seja, sempre que uma alteração relevante estiver funcionando corretamente)
- Não se deve submeter um documento ou código para outras pessoas quando ele possa atrapalhar o trabalho de desenvolvimento do projeto (por exemplo, código que não compila ou com algum erro de execução)
- É importante usar no commit mensagens que descrevam suficientemente a alteração realizada, de modo a evitar que os demais membros do projeto precisem consultar o código diretamente para entender a modificação

Bibliografia e materiais recomendados

- *Version control concepts and best practices*
  `http://homes.cs.washington.edu/~mernst/advice/version-control.html`
- Manual do Git
  `http://git-scm.com/`
- Um tutorial para o Git sem "firulas":
  `http://rogerdudler.github.io/git-guide/index.pt_BR.html`
- Wikipédia
  `http://pt.wikipedia.org/wiki/Sistema_de_controle_de_vers%C3%A3o`