Sistemas de Controle de Versão
~Subversion~

Juliano Ferraz Ravasi
UNESP Rio Claro - 2005

juliano.info

Por que Controle de Versão?

- Programar...
  - ... é difícil;
  - ... toma muito tempo;
  - ... exige cooperação de várias pessoas;
  - ... exige organização.
- Software é algo caro de ser produzido.
- É importante armazenar tudo que é feito.

juliano.info

O que é Controle de Versão?

- Armazenar e ordenar diversas versões de um documento ou projeto
  - Manter um histórico das mudanças efetuadas:
    - Quem?
    - Quando?
    - O quê?
    - Por quê?
  - Analisar as diferenças entre as versões
  - Permitir trabalho cooperativo

juliano.info

O que é Controle de Versão?

- Várias siglas:
  - VCS – *Version Control System*
  - RCS – *Revision Control System*
    - É também o nome de um programa
  - SCM – *Software Configuration Management*
    - É um conceito diferente, mais abrangente

- Controle de versão não precisa necessariamente ser aplicado a softwares.

juliano.info

Vantagens

- Armazenamento centralizado
- Histórico universal de revisões
- Análise de diferenças entre versões
- Trabalho cooperativo
- Ramificação

juliano.info

Vantagens

- **Armazenamento centralizado**
  - Distribuição
  - Cópias de segurança
  - Controle de acesso
- Histórico universal de revisões
- Análise de diferenças entre versões
- Trabalho cooperativo
- Ramificação

juliano.info

Vantagens

- Armazenamento centralizado
- **Histórico universal de revisões**
  - Toda alteração realizada é registrada
    - Quem, quando, o que e por quê.
  - Nada é perdido para sempre
  - Rápido acesso a versões anteriores
- Análise de diferenças entre versões
- Trabalho cooperativo
- Ramificação

juliano.info

Vantagens

- Armazenamento centralizado
- Histórico universal de revisões
- **Análise de diferenças entre versões**
  - Comparar linha-a-linha diferentes versões de um documento, realçando as diferenças
  - Acusar quem foi o último a alterar ou inserir determinada linha
- Trabalho cooperativo
- Ramificação

juliano.info

Vantagens

- Armazenamento centralizado
- Histórico universal de revisões
- Análise de diferenças entre versões
- **Trabalho cooperativo**
  - Vários desenvolvedores trabalhando sobre o mesmo conjunto de arquivos
  - Mesclagem das alterações dos diversos desenvolvedores
- Ramificação

juliano.info

Vantagens

- Armazenamento centralizado
- Histórico universal de revisões
- Análise de diferenças entre versões
- Trabalho cooperativo
- **Ramificação**
  - Fazer alterações em uma cópia do documento, sem alterar o original
  - O documento original pode continuar recebendo outras alterações
  - Combinar (mesclar) as alterações da ramificação com as do documento original

juliano.info

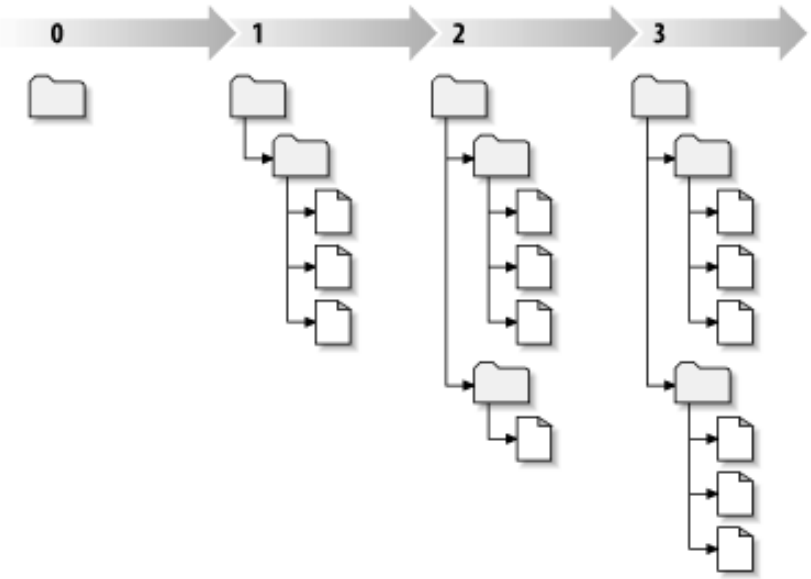
Conceitos

- Repositório
  - Onde são armazenados seus arquivos
  - Possui uma “linha do tempo” embutida
- Cópia de trabalho (*working copy*)
  - Cópia do conteúdo do repositório em um determinado tempo (uma determinada revisão)
  - Onde você faz as alterações nos seus arquivos

juliano.info

Conceitos

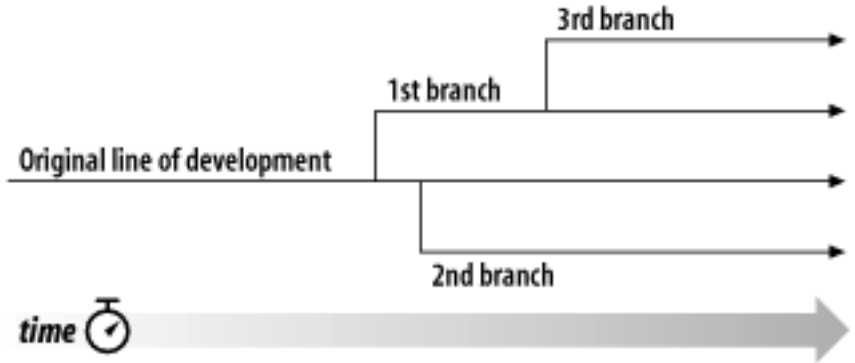
- Revisões (*revisions*)
  - Identificação (geralmente numérica) do estado do repositório (ou de um arquivo) em determinado tempo
- Tronco (*trunk*)
  - Linha onde ocorre o desenvolvimento principal do projeto



juliano.info

Tronco, ramificações e rótulos

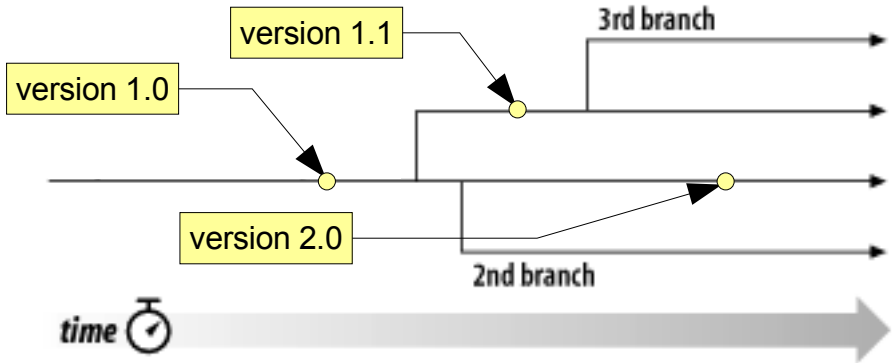
- Ramificações (*branches*)
  - Bifurcações do tronco (ou de outra ramificação)
  - Se tornam independentes, mas compartilham o mesmo histórico
  - Alterações em uma ramificação podem ser recombinadas ao tronco ou a outra ramificação



juliano.info

Tronco, ramificações e rótulos

- Rótulos (*tags*)
  - Marca atribuída ao estado do repositório em determinado tempo
  - Ajuda a encontrar revisões específicas no repositório

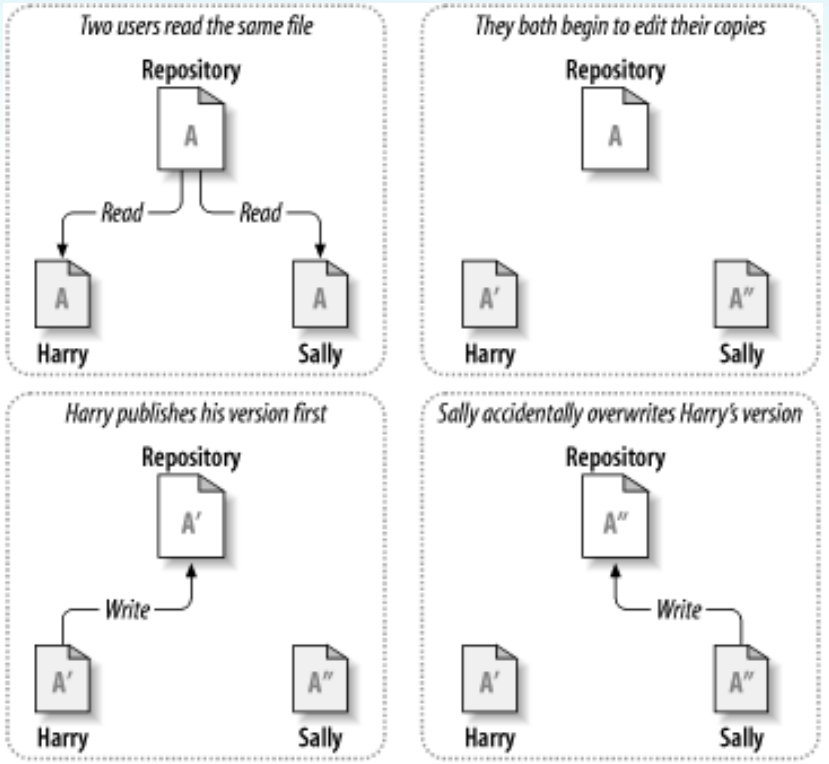


juliano.info

Modelos de Controle de Versões

- Compartilhamento total

Harry e Sally extraem o mesmo arquivo do repositório e fazem mudanças sobre ele. Harry publica sua versão primeiro. Quando Sally publicar a sua versão, ela irá acidentalmente sobrepor a versão de Harry.



juliano.info

Modelos de Controle de Versões

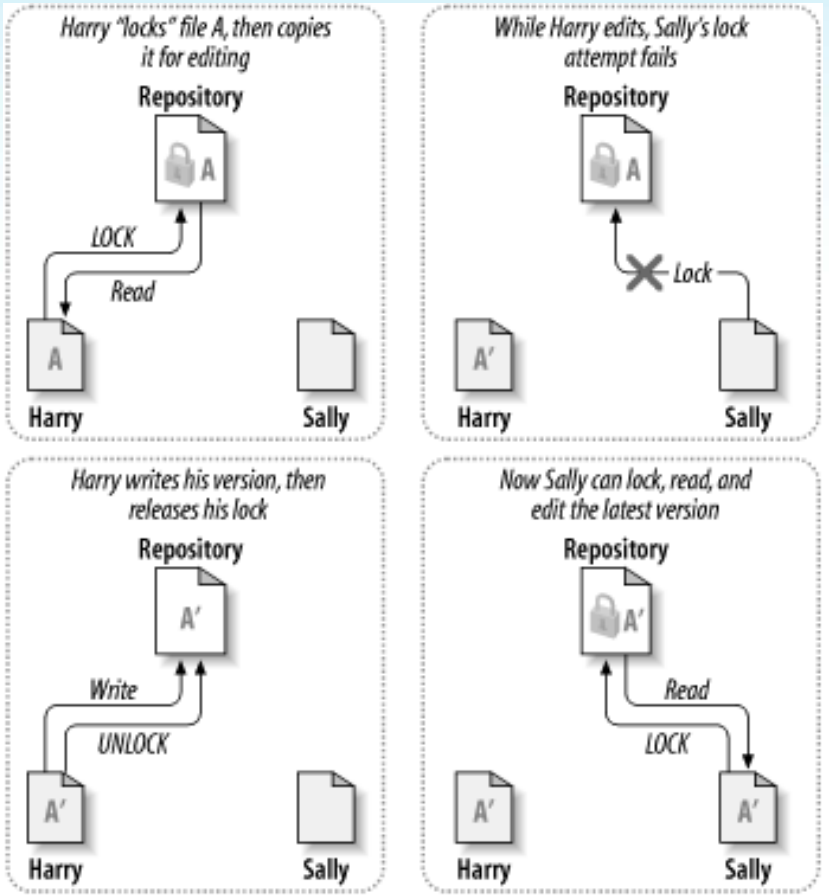
- Compartilhamento total
  - Problemas:
    - Imensa falta de sincronismo
    - Em caso de concorrência, as alterações feitas pela primeira pessoa são simplesmente perdidas

juliano.info

Modelos de Controle de Versões

- lock-modify-unlock

Harry obtém uma “trava” para o arquivo que será editado, e copia para a edição. Sally não consegue travar o arquivo, pois ele já está travado por Harry. Harry submete suas mudanças e libera a trava, Sally pode então travar o arquivo para fazer suas mudanças.



juliano.info

Modelos de Controle de Versões

- lock-modify-unlock
  - Problemas:
    - Duas pessoas não podem trabalhar sobre o mesmo arquivo ao mesmo tempo
    - Podem ocorrer perdas de travas, exigindo manutenção do repositório
    - Falsa sensação de segurança

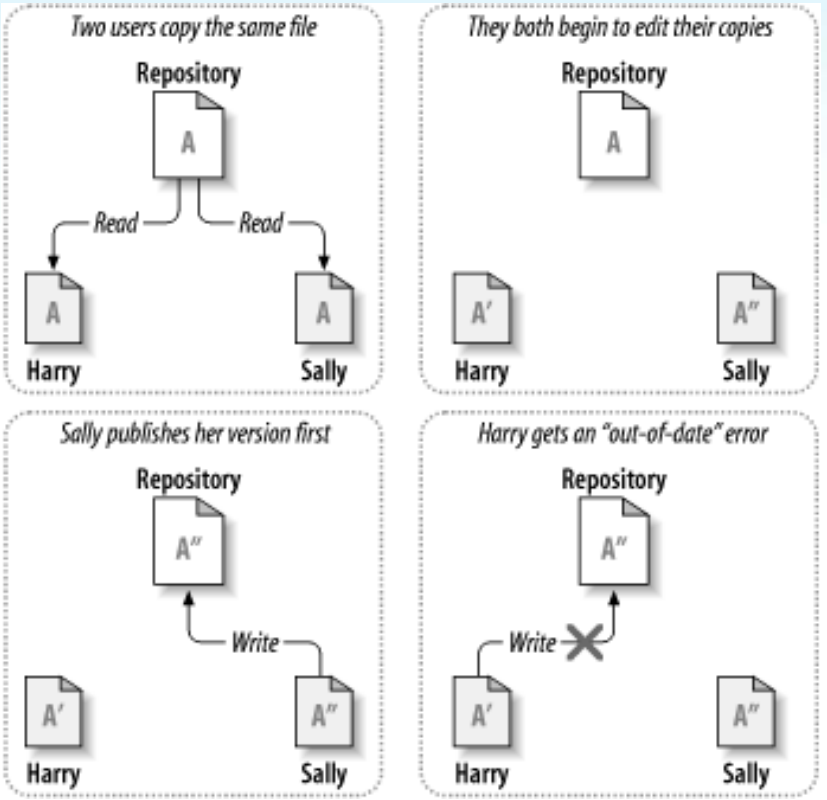
juliano.info

Modelos de Controle de Versões

- copy-modify-merge

Harry e Sally copiam o arquivo para edição, e ambos alteram suas cópias do arquivo original.

Sally submete sua versão para o repositório primeiro. Quando Harry tentar submeter a sua versão, ele obterá um erro, pois suas alterações foram realizadas numa versão "obsoleta" do arquivo.



juliano.info

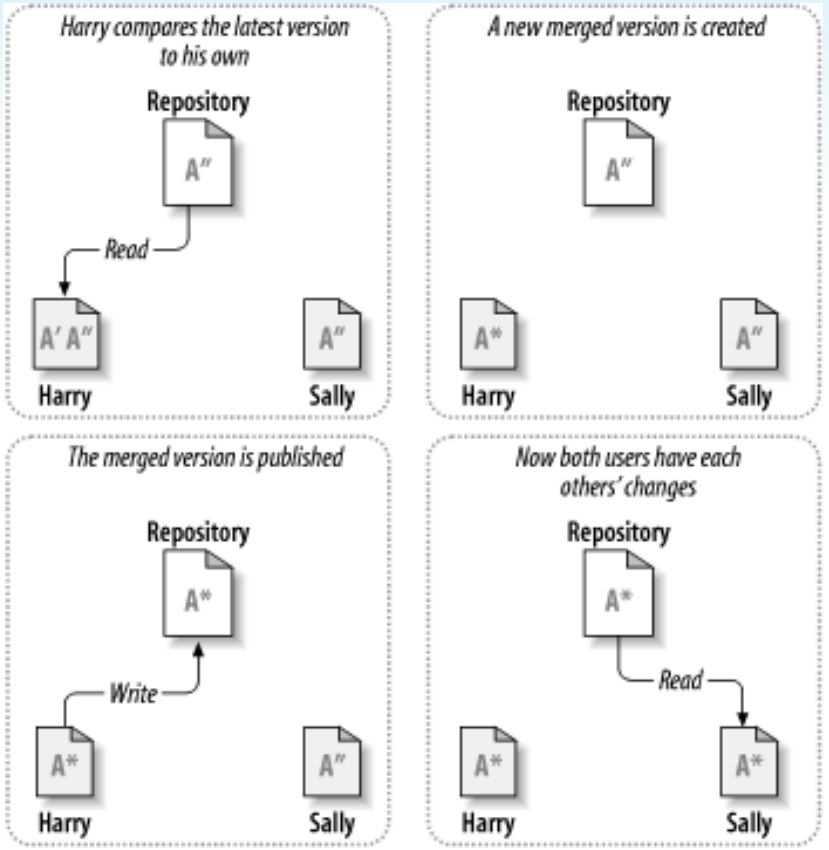
Modelos de Controle de Versões

- copy-modify-merge

O sistema de controle carrega a nova cópia do arquivo do repositório, permitindo que Harry compare suas alterações com as que foram feitas no repositório.

Uma nova versão combinada do arquivo é criada, e então enviada novamente ao repositório.

Sally carrega a nova versão do arquivo do repositório, e ambos terão as alterações um do outro.



juliano.info

Modelos de Controle de Versões

- copy-modify-merge
  - Problema:
    - Conflitos precisam ser tratados manualmente
      - Só ocorrem se ambos alterarem a mesma **parte** do arquivo
      - O sistema oferece ferramentas para ajudar nesta tarefa

juliano.info

O que armazenar?

- O que armazenar no controle de versões?
  - Tudo que é produzido manualmente:
    - código-fonte
    - scripts
    - documentação escrita
    - figuras, imagens e ícones
      - possivelmente apenas os originais, usando ferramentas automáticas para converter formatos ou tamanhos
    - makefiles
      - a menos que sejam criados por um processo automático (./configure nos Unices)

juliano.info

O que armazenar?

- O que **não** armazenar no controle de versões?
  - Arquivos gerados por processos automáticos:
    - código-objeto
    - programas compilados
    - documentação produzida automaticamente
  - Arquivos com configurações locais:
    - nome e senha de acesso a bancos-de-dados
  - Arquivos criados acidentalmente:
    - core dumps
    - arquivos temporários

juliano.info

Sistemas de Controle de Versão

- Centralizados
  - **Subversion**
  - **CVS**
  - Superversion (Java)
  - Stellation
  - RCS (morto)
  - Visual SourceSafe
  - ...
- Distribuídos
  - GNU Arch
  - Monotone
  - Aegis
  - SVK (Subversion)
  - BitKeeper
  - Rational ClearCase
  - ...

juliano.info

CVS e Subversion

- São os dois sistemas open-source mais usados hoje em dia
- CVS (*concurrent versions system*):
  - Criado para substituir o RCS (1980s)
  - Possui uma grande base de usuários
  - Possui defeitos e limitações que são inerentes ao seu design
  - Desenvolvimento quase estagnado

juliano.info

CVS e Subversion

- Subversion:
  - Criado para substituir o CVS
  - Relativamente recente
    - Versão 1.0: 23 de fevereiro de 2004
    - Versão 1.1: 29 de setembro de 2004
    - Versão 1.2: 23 de maio de 2005
  - Quase todos os recursos do CVS...
    ...e quase nenhum de seus defeitos
  - Desenvolvimento extremamente ativo
  - Novo, porém já é maduro
    - Se auto-hospedou antes de atingir a versão alfa (2001)

juliano.info

CVS e Subversion

- Limitações do CVS
  - Diretórios não são “versionados”
  - Não gerencia cópias e moções de arquivos
    - um arquivo renomeado é considerado novo
  - Revisões independentes para cada arquivo
    - não há atomicidade
    - alto custo para a ramificação
  - Cada nome é único no repositório
    - um novo arquivo com o nome de um arquivo apagado herda todo o histórico do arquivo anterior

juliano.info

CVS e Subversion

- Destaques do Subversion
  - Maior consistência entre os seus recursos
    - possui uma curva de aprendizagem menos íngreme
  - Uma revisão para o repositório todo
    - permite submissões atômicas
  - Sistema de arquivos baseado em ligações
    - cópias, moções e renomeações de arquivos e diretórios são suportadas e preservam o histórico
    - cópias, ramificações e rotulações são leves e rápidas

juliano.info

CVS e Subversion

- Destaques do Subversion
  - Permite maior trabalho local (sem precisar acessar o repositório)
    - guarda uma cópia "pura" do diretório de trabalho
  - Meta-dados, incluídos no controle de versões
    - guarda dados extras sobre cada arquivo
  - Escolha do método de armazenamento
  - Escolha do protocolo de transmissão

juliano.info

Subversion em Ação

juliano.info

Estrutura de Diretórios

- No Subversion, tronco, ramificações e rótulos são tratados da mesma forma consistente
  - O sistema de arquivos não faz distinção
  - Você deve criar diretórios para separá-los
  - Estrutura de diretórios sugerida:

```
/                raiz do repositório
    trunk/       tronco do projeto
    branches/    ramificações
    tags/        rótulos
```

  - O desenvolvimento deve ficar no diretório *trunk*
  - Pode-se adotar outros esquemas, a critério do usuário

juliano.info

Criando um Repositório

- **`svnadmin create`** *`diretório`*
  - Este comando cria um diretório com o nome dado contendo um repositório subversion vazio, começando na revisão zero
  - Não é uma cópia de trabalho
  - Você deve extrair uma cópia de trabalho desse repositório e criar alguns diretórios básicos (*trunk*, *branches* e *tags*)

juliano.info

Checkout

- Extrai uma cópia de trabalho do repositório
  - Por padrão, extrai a última revisão
  - O repositório não precisa estar no mesmo sistema
- **`svn checkout`** *`file:///end/do/repos`* *`dir`*
  - Um diretório *dir* é criado no diretório corrente, contendo sua cópia de trabalho
  - Para criar os diretórios básicos:
    - `cd` *`dir`*
    - **`svn mkdir`** `trunk branches tags`
    - **`svn commit`** `-m "Estrutura inicial"`

juliano.info

Checkout

- Todas as formas de acessar um repositório:
  - *file:///endereço/do/repositório*
    - repositório local
  - *http://servidor/endereço/do/repositório*
  - *https://servidor/endereço/do/repositório*
    - repositório externo, via WebDAV sobre HTTP
  - *svn://servidor/endereço/do/repositório*
    - repositório externo, através do protocolo *svnserve*
  - *svn+ssh://servidor/endereço/do/repositório*
    - idem ao *svn://*, porém, sobre o protocolo SSH

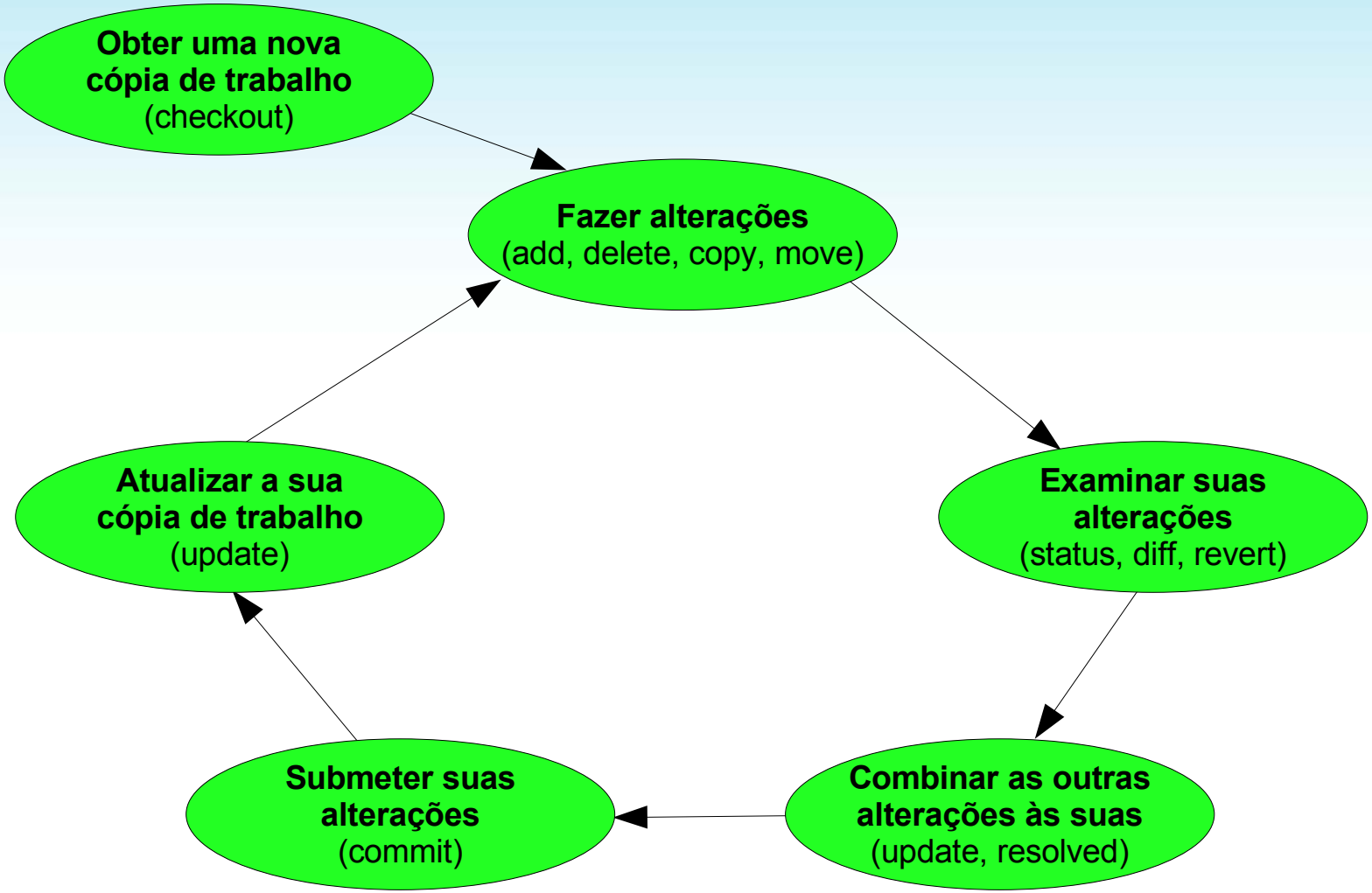
juliano.info

Checkout

- Depois que a estrutura inicial tiver sido criada, não há necessidade de extrar uma cópia completa, o diretório *trunk* é suficiente
  - **`svn checkout`** *`file:///end/do/repos/`***`trunk`** *`dir`*

juliano.info

Ciclo Básico de Trabalho



juliano.info

Trabalhando na sua Cópia de Trabalho

- O Subversion fornece comandos específicos para alterar arquivos na cópia de trabalho
  - **`svn add`** *`arquivo`* **`...`**
    - torna o *arquivo* parte da cópia de trabalho e agenda sua adição ao repositório no próximo *commit*
  - **`svn delete`** *`arquivo`* **`...`**
    - apaga o *arquivo* da cópia de trabalho e agenda sua "remoção" do repositório no próximo *commit*
  - **`svn copy`** *`origem destino`*
    - cria uma cópia do arquivo ou diretório *origem* para o *destino*, herdando o histórico de alterações do original

juliano.info

Trabalhando na sua Cópia de Trabalho

- **`svn move`** *`origem destino`*
  - move ou renomeia o arquivo ou diretório *origem* para *destino*, preservando seu histórico de alterações
- **`svn mkdir`** *`diretório`* `...`
  - cria um novo diretório vazio
- **`svn rmdir`** *`diretório`* `...`
  - apaga o diretório especificado
- **`svn revert`** `[`*`arquivo-ou-diretório`* `...]`
  - reverte as alterações realizadas na cópia de trabalho, retornando ao que estava no repositório no momento do *checkout* (ou *update*)

juliano.info

Examinando as suas Alterações

- **`svn status`** [*`arquivo-ou-diretório`* `...`]
  - diz quais arquivos foram adicionados, apagados ou alterados na cópia de trabalho, e que diferem do repositório
  - alguns códigos de status:
    - `'A': item adicionado`
    - `'C': item em conflito`
    - `'D': item apagado`
    - `'M': item modificado`
    - `'?': item desconhecido`
    - consulte outros códigos no *svnbook*

juliano.info

Examinando as suas Alterações

- **`svn diff`** [*arquivo-ou-diretório* ...]
  - mostra as diferenças, linha-a-linha, entre a cópia de trabalho e o repositório
  - com o parâmetro **`-r M[:N]`**, mostra as diferenças entre as revisões *M* e *N* (ou entre *M* e a cópia de trabalho)

```
Index: bar.c
===================================================================
--- bar.c       (revision 3)
+++ bar.c       (working copy)
@@ -1,7 +1,12 @@
+#include <sys/types.h>
+#include <sys/stat.h>
+#include <unistd.h>
+
+#include <stdio.h>

 int main(void) {
-  printf("Sixty-four slices of American Cheese...\n");
+  printf("Sixty-five slices of American Cheese...\n");
 return 0;
 }
```

juliano.info

Atualizando a sua cópia e submetendo as suas alterações

- **`svn update`** [*`arquivo-ou-diretório`* `...`]
  - atualiza a cópia de trabalho com a revisão mais recente do reposiório; podem surgir conflitos
- **`svn commit`** [*`arquivo-ou-diretório`* `...`]
  - submete as suas alterações na cópia de trabalho para o repositório; você deverá informar uma mensagem de *log* que será registrada junto à submissão, explicando o que foi alterado

juliano.info

Resolvendo Conflitos

- Conflitos ocorrem quando duas pessoas submetem alterações para o mesmo arquivo
  - Cenário de exemplo:
    - Harry e Sally extraem a revisão 1 do arquivo *sandwich.txt* e cada um começa a fazer as suas alterações.
    - Sally submete primeiro as suas alterações do arquivo, e essa nova revisão recebe o número 2.
    - Harry atualiza sua cópia de trabalho, e as alterações que Sally fez em *sandwich.txt* entram em conflito com as suas.

juliano.info

Resolvendo Conflitos

- Quando um conflito ocorre, se as alterações
  - forem em regiões distintas do arquivo:
    - o Subversion combina as alterações automaticamente
    - mostra a letra ‘G’ (*mer**g**ed*) durante o *update*
  - forem na mesma região do arquivo:
    - o Subversion cria 3 arquivos no diretório:
      - “arquivo.mine” sua versão do arquivo
      - “arquivo.rOLD” arquivo como estava na revisão *OLD*
      - “arquivo.rNEW” arquivo como estava na revisão *NEW*
    - mostra a letra ‘C’ (***c**onflict*) durante o *update*
    - trava a cópia de trabalho em estado de conflito

juliano.info

Resolvendo Conflitos

- Quando houver um conflito não-solucionado:
  - a cópia de trabalho não poderá ser submetida até que o conflito seja solucionado
  - o usuário deve comparar os 3 arquivos gerados pelo Subversion e combinar suas alterações
  - o arquivo, se for do tipo texto, conterá as alterações conflitantes demarcadas, para que você possa facilmente encontrá-las e resolvê-las

```
Top piece of bread
Mayonnaise
Lettuce
Tomato
Provolone
<<<<<<< .mine
Salami
Mortadella
Prosciutto
=======
Sauerkraut
Grilled Chicken
>>>>>>> .r2
Creole Mustard
Bottom piece of bread
```

juliano.info

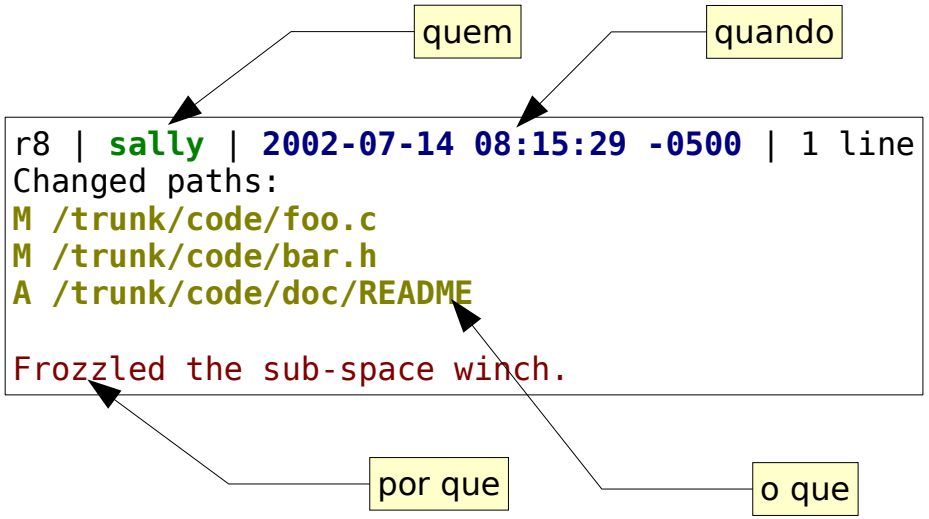
Resolvendo Conflitos

- Depois que o conflito for resolvido:
  - **`svn resolved`** *`arquivo`*`...`
    - informa ao Subversion que você já resolveu o conflito que fora detectado no arquivo
    - ao executar este comando, o Subversion apaga os arquivos '*.mine*', '*.rOLD*' e '*.rNEW*' criados durante o *update*
    - se não houver mais conflitos, a cópia de trabalho é liberada para que seja submetida ao repositório
- A melhor forma de resolver conflitos e não criá-los, ou seja, conversando!

juliano.info

Obtendo informações da sua Cópia de Trabalho

- **`svn info`** [*`arquivo-ou-diretório ...`*]
  - fornece informações sobre o estado do arquivo ou diretório especificado no controle de versões
- **`svn log`** [*`arquivo-ou-diretório ...`*]
  - mostra o histórico de alterações registradas no repositório (responde às 4 perguntas: quem, quando, o que e por quê)



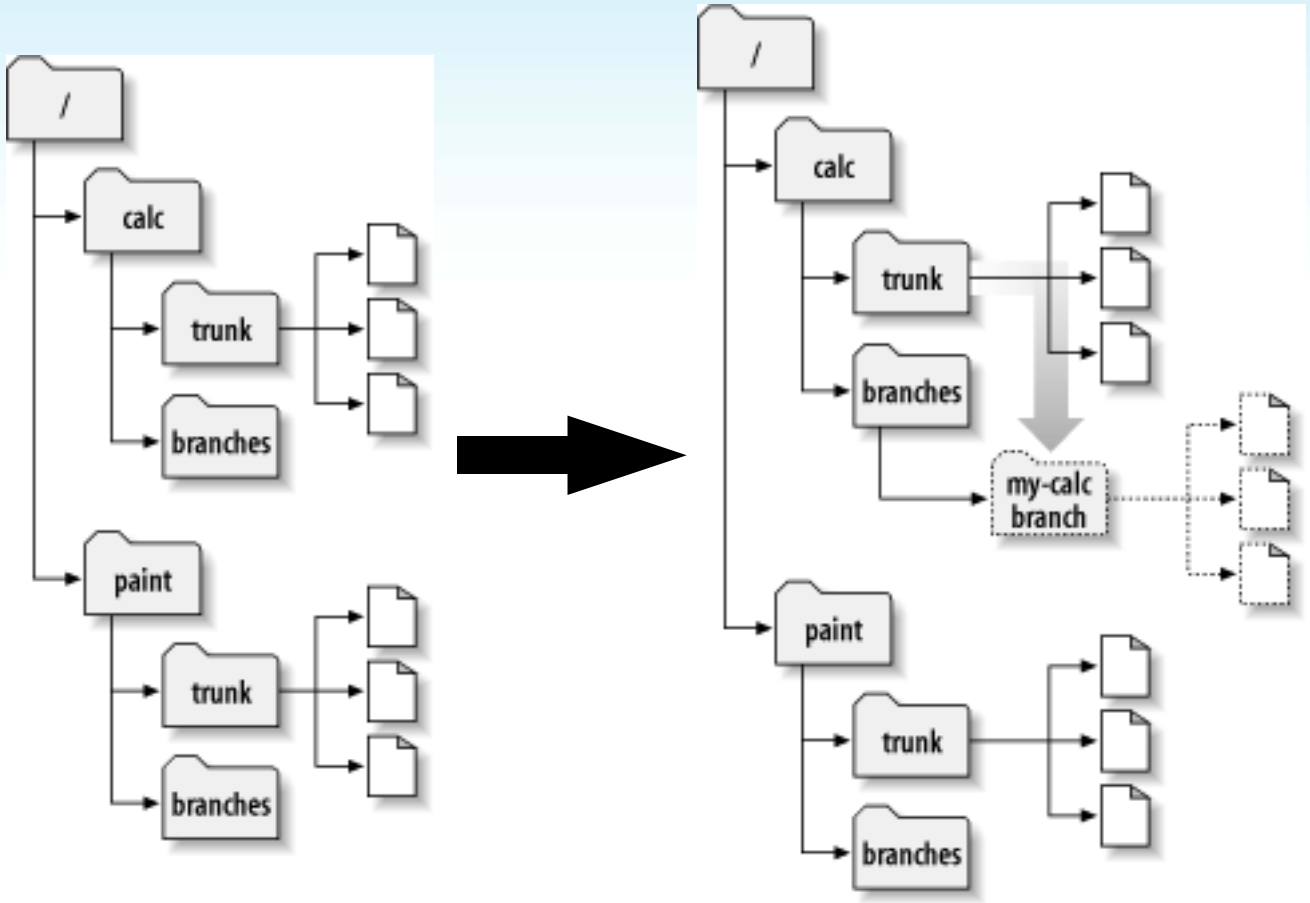
juliano.info

Ramificações

- Ramificações são úteis quando deseja-se:
  - implementar recursos complexos
  - fazer grandes alterações no projeto
  - efetuar correções a versões antigas do projeto
  - criar uma versão especial do projeto
- No Subversion, o processo de ramificação consiste em:
  - criar uma cópia do diretório *trunk* para um diretório dentro de *branches*
  - extrair uma cópia da ramificação para trabalhar

juliano.info

Ramificações



juliano.info

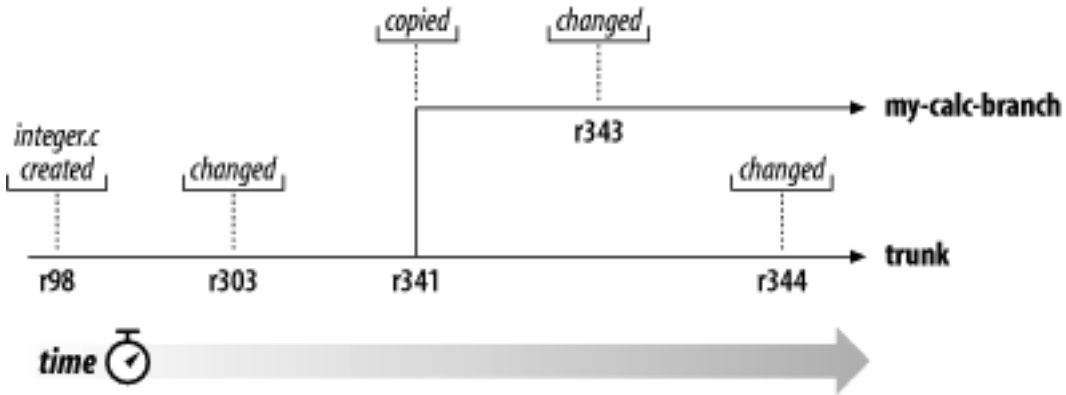
Ramificações

- Ramificando o tronco do projeto
  - **`svn copy`** *`file:///.../repos/`***`trunk`**
    *`file:///.../repos/`***`branches/mybranch`**
    - Cria uma ramificação (na verdade, uma cópia) do tronco.
    - Da mesma forma, pode-se ramificar outra ramificação.
- Trabalhando na ramificação
  - **`svn checkout`**
    *`file:///.../repos/`***`branches/mybranch`**
    - É um *checkout* comum, só que agora nós solicitamos uma cópia da ramificação, ao invés do tronco.

juliano.info

Ramificações

- Lembre-se: O Subversion faz cópias **leves**.
  - A ramificação consiste em criar uma ligação para o tronco em uma determinada revisão.
  - Todo o histórico do tronco é preservado na ramificação.
  - Não se preocupe, o tamanho do repositório não cresce a cada ramificação.



juliano.info

Ramificações

- Para combinar alterações de uma ramificação para outra:
  - Na cópia de trabalho que receberá as alterações:
  - **`svn merge`** `-r` *`M`*`:`*`N`* *`file:///.../repos/`***`trunk`**
    - *M* e *N* são os números das revisões que compreendem as alterações a serem combinadas.
    - Se algo der errado, você tem a opção de fazer um *revert* antes de submeter.
    - Da mesma forma que o *update*, o *merge* pode gerar conflitos.
    - É possível usar o *merge* para combinar "ao contrário", ou seja, desfazer mudanças no repositório.

juliano.info

Ramificações

- Alternando a cópia de trabalho
  - **`svn switch`** *`file:///.../repos/`***`branches/mybranch`**
    - Alterna a cópia de trabalho para a ramificação especificada.
    - Da mesma forma, pode ser usado para alternar devolta ao tronco.
    - Este procedimento é muito semelhante ao *update*.

juliano.info

Rótulos

- Rótulos são úteis para:
  - demarcar pontos-chave do desenvolvimento
  - armazenar as diversas versões do projeto

- Nenhuma novidade:
  - `svn copy` *file:///.../repos/**trunk***
    *file:///.../repos/**tags/version-1.0***

juliano.info

Rótulos

- Ramificações *vs.* Rótulos
  - Ao contrário das ramificações, os rótulos não devem ser usados para desenvolvimento.
  - Devem ser “somente-leitura” depois de criados.
  - O repositório pode ser configurado para proteger os rótulos contra modificações, depois de criados.

juliano.info

E agora?

- Ajuda *online*:
  - **`svn help`** [*comando*]
  - **`svn`** *comando* **`--help`**

- Version Control with Subversion (*svnbook*):
  - http://svnbook.red-bean.com/
  - http://svnbook.org/

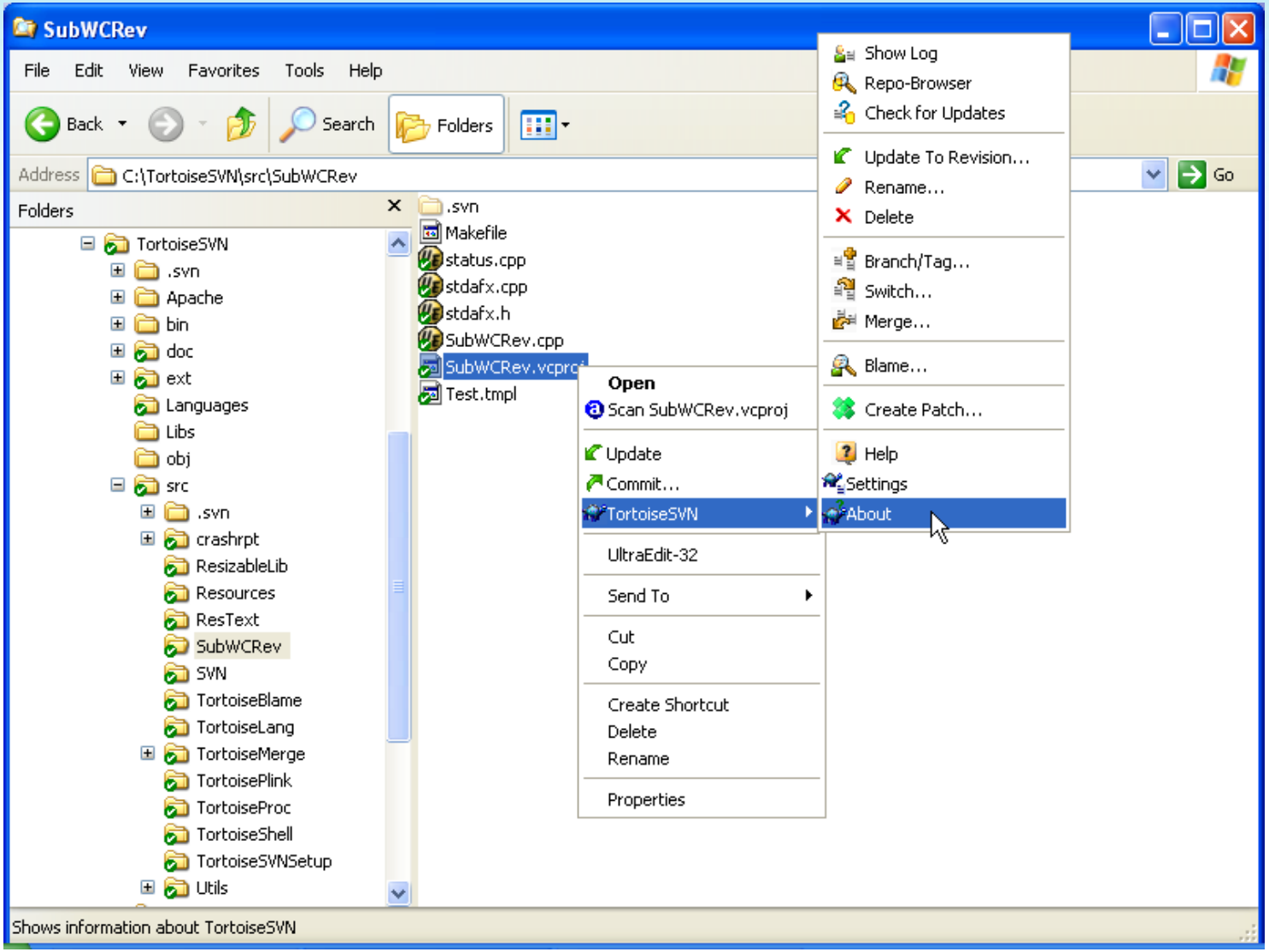
juliano.info

Ferramentas

juliano.info

TortoiseSVN

http://tortoisesvn.tigris.org/



juliano.info

Referências

- Subversion
  - http://subversion.tigris.org/
  - http://svnbook.red-bean.com/
- TortoiseSVN
  - http://tortoisesvn.tigris.org/
- CVS
  - http://www.cvshome.org/
  - http://cvsbook.red-bean.com/

juliano.info

Obrigado!

- Juliano Ferraz Ravasi
  - GBIRC – UNESP Rio Claro – 2005
  - Contato:
    - http://juliano.info/
    - contact@juliano.info

"Uma vela não perde nada acendendo outra vela."

– Erin Majors

juliano.info