UNISINOS

CVS

Concurrent Versions System

Ricardo Nabinger Sanchez
rnsanchez@gmail.com

http://cscience.org/~rnsanchez/cvs

Roteiro

- Utilização Básica
  - Conceitos
  - Criando Repositórios e Módulos
  - Retirada
  - Atualização
  - Adição e Remoção
  - Pânico!
  - Referências
- Utilização Avançada
  - Histórico e Log
  - Repositórios Diferentes e Remotos
  - Status
  - Marcas (Tags)
  - Ramificação (Branches)
  - União
  - Conflitos
  - Exportar
  - Patches
  - Atalhos

Minicurso de CVS? (1)

- CVS é um acrônimo para "Concurrent Versions System", ou Sistema de Versões Concorrentes
- CVS é uma ferramenta livre e de código aberto para desenvolvimento colaborativo, madura e muito utilizada
- Pode ser usada em diversas áreas onde se deseja utilizar desenvolvimento colaborativo:
  - Software
  - Documentação
  - Qualquer outra área onde se manipule arquivos com "texto em claro", ou seja, formato não binário

Minicurso de CVS? (2)

- CVS é uma ferramenta bastante poderosa, mas requer um certo tempo para ser dominada
- Sua utilização básica é razoavelmente simples, e existe muita documentação disponível (especialmente na Internet)
- O controle de versões é uma extensão natural do processo de desenvolvimento, agregando qualidade de processo
- O CVS gera dados de sua utilização, que podem ser usados para criar relatórios diversos, como por exemplo o tempo (com grande precisão) necessário para uma determinada implementação ser concluída

Controle de Versão

- Também conhecido como “Controle de Revisão”, ou apenas “Revisão”
- Alterações sucessivas em um arquivo são identificadas por um número ou letra, incrementados a cada revisão submetida:

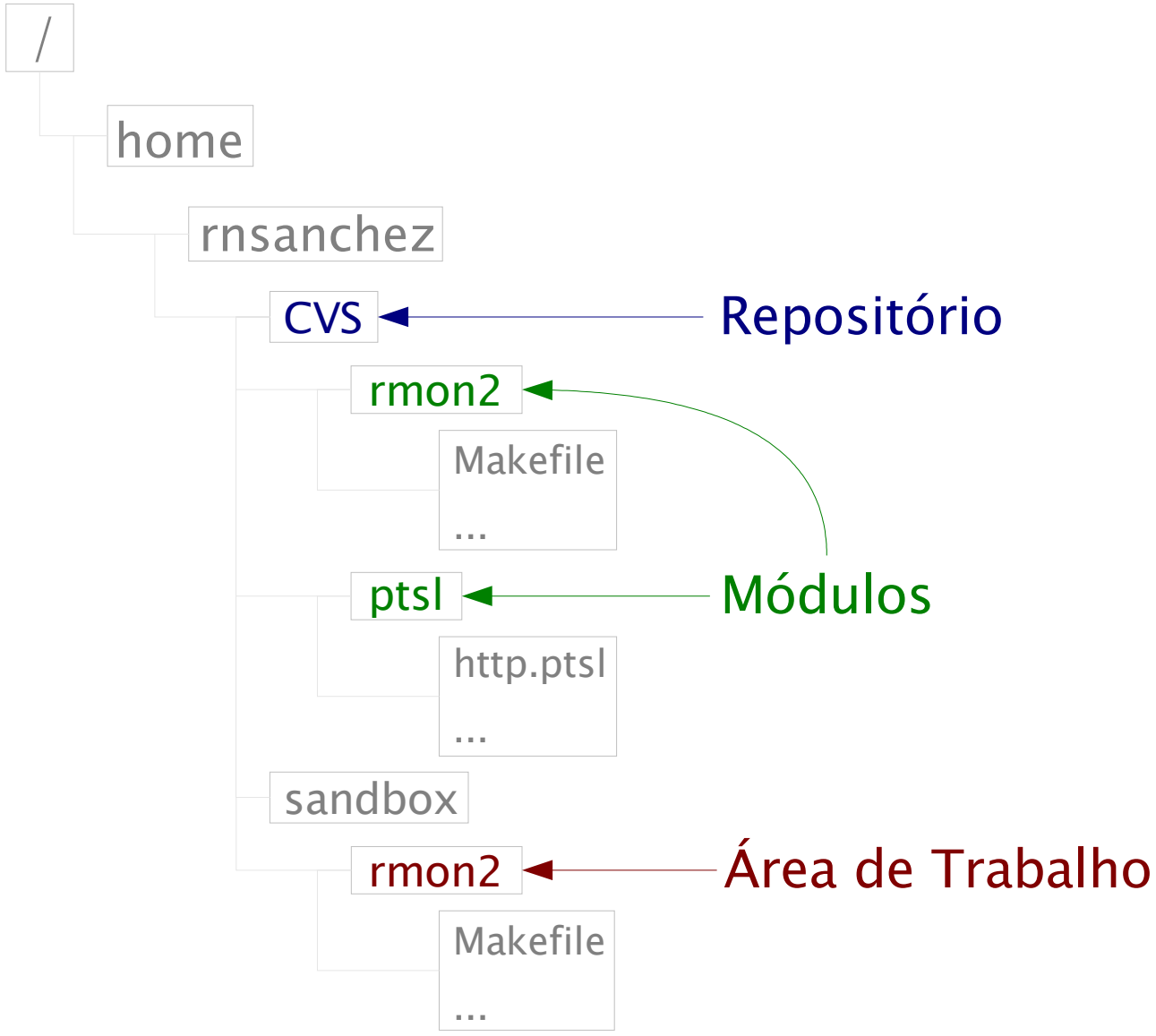
Arquivo	Revisão	Data
hello.c	0.0	10/Out/2004
hello.c	0.1	11/Out/2004
hello.c	0.2	14/Out/2004

- Os números de revisão são apenas para uso interno do CVS e não indicam a versão do projeto (ex: `linux-2.6.9`)
- Cada revisão mantém o estado do arquivo em um dado momento, e qualquer revisão poderá ser acessada

Conceitos Básicos

- Repositório – local onde ficam armazenados os arquivos que compõem os diversos módulos. Um repositório pode ser local (armazenado na própria estação de trabalho, em um determinado diretório) ou em um servidor remoto
- Módulo – separação lógica dos arquivos de diferentes projetos, sob um mesmo repositório
- Área de trabalho – local onde o desenvolvedor mantém os arquivos de um determinado módulo para modificá-los

Esquema de um Repositório Local



Antes de Começar...

- Para o minicurso, assumiremos um repositório local em ambiente GNU/Linux, usando shell Bash ou Zsh
- A criação de um repositório local possui os seguintes requisitos:
  - Um diretório (vazio) que o usuário possa usar:

```
$ mkdir /home/rnsanchez/CVS
```

  - Redefinir a variável de ambiente CVSROOT para apontar para este diretório:

```
$ export CVSROOT=/home/rnsanchez/CVS
```

Inicializando o Repositório

- A inicialização do repositório consiste em instruir ao CVS para criar um conjunto de arquivos de administração, para que se possa adicionar módulos em seguida:

```
$ cvs init
```

- Após este comando, o CVS criará um diretório chamado CVSROOT contendo os arquivos de que necessita para gerenciar o repositório (arquivos administrativos)
- O repositório já se encontra em condições de criar módulos e importar os respectivos arquivos

Criando um Módulo (1)

- Para criar um módulo no repositório, basta importar os arquivos desejados a partir do diretório raiz do projeto
- Diretório raiz de um projeto é o diretório que contém todos os arquivos de um determinado projeto:

```
rnsanchez@sauron:~/rmon2$ ls -F
doc/  etc/  include/  net-snmp/  src/  COPYING
Doxyfile  ERRORS  INSTALL  Makefile  README
```

- Arquivos que não devem ser importados (arquivos temporários ou de compilação, por exemplo) precisam ser removidos antes da importação

Criando um Módulo (2)

- Criar o módulo rmon2 importando os arquivos:

```
$ cvs import rmon2 rnsanchez inicial
N rmon2/COPYING
N rmon2/Doxyfile
N rmon2/Makefile
...
N rmon2/src/trasserlib.h

No conflicts created by this import
```

- O N no início de cada linha indica que o arquivo importado é um arquivo novo, que ainda não existia no módulo

Criando um Módulo (3)

- Sintáxe do comando:

```
$ cvs import  módulo  vendor_tag  release_tag
$ cvs import  rmon2   rnsanchez   inicial
```

  - `módulo` indica o módulo que deve ser criado no repositório, se este não ainda existir
  - `vendor_tag` geralmente indica o distribuidor ou autor do projeto
  - `release_tag` indica o marcador que deve ser usado durante a importação dos arquivos

- Após a importação, é costume remover os arquivos importados. Os precavidos fazem backup.

Informações do CVS

- O CVS possui algumas palavras-chaves que, quando detectadas por ele, são substituídas por informações úteis:
  - `$Id$` é a mais usada, sendo substituída por um cabeçalho com diversas informações juntas:

    ```
    $Id: Makefile,v 0.3 2004/09/27 14:09:02 rnsanchez Exp $
    ```

  - `$Author$` é substituída pelo login do usuário que modificou por último o arquivo
  - `$Date$` é substituída pela data e hora (UTC) da entrega do arquivo
  - `$Log$` é substituída pelas mensagens que os desenvolvedores inserem quando enviam as alterações para o repositório
  - Existem outras, mas essas são as mais legais

Cópia de Trabalho de um Módulo

- Para modificar os arquivos de um módulo, devemos antes fazer uma cópia (checkout) do mesmo para a área de trabalho
- No exemplo, o diretório ~/sandbox foi escolhido como área de trabalho, a cópia do módulo será feita lá:

```
rnsanchez@sauron:~/$ cd ~/sandbox
rnsanchez@sauron:~/sandbox$ cvs checkout rmon2
...
U rmon2/src/trasserlib.h
```

- O código U indica que o CVS atualizou o arquivo local com uma nova cópia do arquivo do repositório

Antes de Submeter Alterações...

- Em um ambiente de desenvolvimento colaborativo, é possível que outra pessoa tenha alterado alguns arquivos do mesmo módulo, e já tenha enviado para o repositório
- Para evitar que possíveis alterações sejam desfeitas, é importante atualizar os arquivos da área de trabalho antes de enviar as alterações para o repositório:

```
rnsanchez@sauron:~/sandbox/rmon2$ cvs update
...
cvs update: Updating rmon2/src
```

- Neste momento pode ocorrer um conflito, que deve ser solucionado editando-se os arquivos envolvidos

Submetendo Alterações

- Realizadas as modificações necessárias, o próximo passo é enviá-las para o repositório com o seguinte comando:

```
rnsanchez@sauron:~/sandbox/rmon2$ cvs commit
...
cvs commit: Examining src
Checking in src/main.c;
/home/rnsanchez/CVS/rmon2/src/main.c,v  <--  main
new revision: 1.2; previous revision: 1.1
done
```

- Estas modificações estarão imediatamente disponíveis para os outros colaboradores (assim que fizerem `cvs update`)

Adicionando Arquivos

- Quando se quer adicionar um arquivo a um módulo, é necessário adicioná-lo explicitamente, pois o CVS não adiciona ou remove arquivos automaticamente:

```
rnsanchez@sauron:~/sandbox/rmon2/src$ cvs add novo.c
cvs add: scheduling file `novo.c' for addition
cvs add: use 'cvs commit' to add this file permanently
$ cvs commit -m “Incluído arquivo que faltava” novo.c
RCS file: /home/rnsanchez/CVS/rmon2/src/novo.c,v
done
Checking in novo.c;
/home/rnsanchez/CVS/rmon2/src/novo.c,v  <--  novo.c
initial revision: 1.1
done
```

Adicionando Arquivos Binários

- Embora arquivos binários não sejam o ponto forte do CVS, é possível incluí-los nos módulos, usando uma variação do comando CVS add:

```
$ cvs add -kb arquivo_binario.pdf
```

- A diferença no tratamento de arquivos binários é que, por exemplo, não será possível verificar as diferenças entre versões de arquivos binários, ainda que o CVS aplique o controle de versões a esses arquivos

Removendo Arquivos

- A remoção de arquivos também deve ser feita de forma explícita. O arquivo deve ser removido da área de trabalho e em seguida atualiza-se o repositório:

```
rnsanchez@sauron:~/sandbox/rmon2/src$ rm fora.c
$ cvs remove fora.c
cvs remove: scheduling `fora.c' for removal
cvs remove: use 'cvs commit' to remove this file permanently
$ cvs commit -m “Arquivo finalmente removido” fora.c
Removing fora.c;
/home/rnsanchez/CVS/rmon2/src/fora.c,v  <--  fora.c
new revision: delete; previous revision: 1.1
done
```

Atualização da Área de Trabalho

- É possível (e importante!) atualizar os arquivos da área de trabalho com as modificações existentes no repositório:
  - Toda a área de trabalho:

```
rnsanchez@sauron:~/sandbox/rmon2$ cvs update
```

  - Apenas um determinado diretório (e sub-diretórios):

```
rnsanchez@sauron:~/sandbox/rmon2/src$ cvs update
```

  - Determinado diretório, exceto sub-diretórios:

```
rnsanchez@sauron:~/sandbox/rmon2$ cvs update -l
```

  - Determinados arquivos:

```
...:~/sandbox/rmon2$ cvs update inc/rmon2.h src/rmon2.c
```

Em Caso de Aperto... (1)

- Para listar os comandos CVS e uma breve descrição deles:

```
$ cvs --help-commands
CVS commands are:
    add          Add a new file/directory to the repository
    admin        Administration front end for rcs
    annotate     Show last revision where each line was modified
    checkout     Checkout sources for editing
    commit       Check files into the repository
    diff         Show differences between revisions
    edit         Get ready to edit a watched file
    editors      See who is editing a watched file
   ...           ...
```

Em Caso de Aperto... (2)

- Para ver a ajuda de um determinado comando:

```
$ cvs --help update
Usage: cvs update [-APCdflRp] [-k kopt] [-r rev] [-D date]
  [-j rev] [-I ign] [-W spec] [files...]
  -A      Reset any sticky tags/date/kopts.
  -P      Prune empty directories.
  -C      Overwrite locally modified files with clean repository
          copies.
  ...     ...
```

- O manual online do CVS possui bastante informação, com a vantagem de ter descrições mais longas (e úteis):

```
$ man cvs
```

Significado dos Códigos de Saída

- U – mais comum, indica que o CVS atualizou (ou criou) um arquivo com uma cópia completa do repositório
- P – indica que o CVS atualizou um arquivo usando patch, ou seja, enviou apenas as diferenças do repositório
- M – indica que o CVS detectou uma alteração na cópia local em relação ao repositório
- C – indica que o CVS encontrou um conflito ao tentar mesclar as atualizações do repositório com a cópia local
- R – arquivo local removido por não fazer mais parte do repositório
- N – arquivo local adicionado ao repositório
- ? – arquivo local que não foi encontrado no repositório, e, portanto, será ignorado

Parte Fácil Concluída!

- Referências recomendadas:
  - Caetano, Cristiano. CVS: Controle de Versões e Desenvolvimento Colaborativo. ISBN 8575220519
  - Neubert, Marden. CVS: Guia de Consulta Rápida. ISBN 857522056X
  - CVS Quick Reference Card. `http://refcards.com/refcards/cvs/index.html`
  - CVS Quick Reference Card. `http://tnerual.eriogerg.free.fr/cvs.html`
  - CVS Home. `http://www.cvshome.org`

UNISINOS

CVS

Concurrent Versions System

Parte Pesada

Histórico de Acesso ao Repositório

- Com o comando history, pode-se verificar como o repositório tem sido acessado:
  - Todos os usuários:
    - `$ cvs history -a`
  - Todos os tipos de ação:
    - `$ cvs history -e`
  - Apenas arquivos alterados por qualquer usuário:
    - `$ cvs history -ca`
  - Todos os acessos de um determinado usuário:
    - `$ cvs history -e -u usuario_desejado`

Histórico de Revisões dos Arquivos (Log)

- O Log do CVS pode apresentar muita informação, especialmente se for executado sem parâmetros (loucura total!)
- Para mostrar o log de alterações entre um determinado par de revisões (numéricas, não simbólicas):

```
$ cvs log -r 0.2.0.2:0.3.0.2 Makefile
...
total revisions: 13;    selected revisions: 1
description:
----------------------------
revision 0.3
date: 2004/09/27 14:09:02;  author: rnsanchez;  state: Exp;  lines: +17
   -13
Unindo revisões da tag rmon2_0_3_o_pre6.
```

Utilizando Repositórios Diferentes (1)

- Eventualmente, um usuário pode querer acessar um repositório diferente do definido na variável $CVSROOT
- Nestes casos, o usuário passa a localização do repositório que deseja acessar (deve ser conhecido de antemão) com o parâmetro -d

```
$ cvs -d /home/outro_usuario/CVSROOT_dele
```

- Se for incômodo usar a opção -d, pode-se redefinir a variável CVSROOT e depois restaurá-la:

```
$ export CVSROOT=~outro_usuario/CVSROOT_dele
...
$ export CVSROOT=~/CVS
```

Utilizando Repositórios Diferentes (2)

- Outra possibilidade é acessar repositórios remotos através de um password server (pserver):

```
$ cvs -d :pserver:anonymous@cvs.sf.net:/cvsroot/gaim co gaim
```

- Também é possível acessar remotamente o repositório de uma estação onde o usuário tenha acesso a shell, usando uma aplicação externa (um túnel SSH, por exemplo):

```
$ export CVS_RSH=/usr/bin/ssh
$ cvs -d :ext:usuario@cvs.sf.net:/cvsroot/gaim co gaim
```

- A variável de ambiente $CVS_RSH controla a aplicação externa que deve ser usada para intermediar a comunicação (/usr/bin/ssh, no caso)

Utilizando Repositórios Diferentes (3)

- Felizmente, a localização do repositório é armazenada quando se faz uma retirada (checkout), facilitando a vida do usuário:

```
$ cvs -d :ext:usuario@cvs.sf.net:/cvsroot/gaim co gaim
$ cd gaim
...
$ cvs update
$ cvs commit
```

- O CVS verifica se no diretório onde o usuário executa os comandos existe o diretório CVS/, que contém informações inclusive sobre onde está o repositório, prevalecendo sobre a variável $CVSROOT

Exibindo Status dos Arquivos (1)

- Para exibir o status de um ou mais arquivos (ou o módulo, se não for incluído nenhum arquivo como parâmetro):

```
$ cd ~/sandbox/rmon2
$ cvs status Makefile
===================================================================
File: Makefile        Status: Up-to-date
Working revision:     0.3
Repository revision: 0.3   /home/rnsanchez/CVS/rmon2/Makefile,v
Sticky Tag:           rmon2_0_3_0_pre7 (branch: 0.3.2)
...
```

Exibindo Status dos Arquivos (2)

- Para exibir o status e tags:

```
$ cd ~/sandbox/rmon2
$ cvs status -v Makefile
===================================================================
File: Makefile          Status: Up-to-date
Working revision:       0.3
Repository revision: 0.3    /home/rnsanchez/CVS/rmon2/Makefile,v
Sticky Tag:             rmon2_0_3_0_pre7 (branch: 0.3.2)
...
Existing Tags:
    rmon2_0_3_0_pre7    (branch: 0.3.2)
    rmon2_0_3_0_pre6    (branch: 0.2.2)
    ...
```

Exibindo Status dos Arquivos (3)

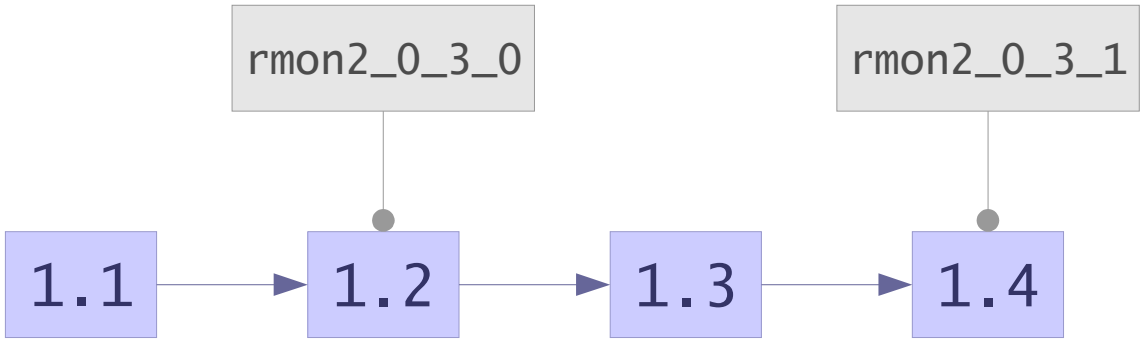
- Mensagens comuns para o status:
  - Up-to-date: arquivo está atualizado
  - Locally Modified: arquivo na área de trabalho foi modificado e ainda não submetido ao repositório (commit)
  - Locally Added/Removed: arquivo adicionado ou removido e ainda não submetido ao repositório (commit)
  - Needs Checkout/Patch: existe uma revisão mais recente do arquivo no repositório do que na área de trabalho (update)
  - Needs Merge: o arquivo possui uma revisão nova no repositório e também foi modificado localmente (update)

Exibindo Status dos Arquivos (4)

- File had conflicts on merge: durante a atualização (update), o CVS constatou um conflito enquanto mesclava as alterações. O usuário deverá corrigir o conflito manualmente, e então submeter o arquivo para o repositório (commit)
- Unknown: o arquivo foi ignorado pelo CVS, pois não foi importado ou adicionado explicitamente, e, portanto, não há qualquer controle de versão sobre ele

Tags (1)

- Tags são usadas para marcar um determinado momento de um arquivo, arquivos ou módulo usando uma referência da escolha do desenvolvedor
- Existem restrições para o nome escolhido:
  - Deve começar com uma letra
  - Não pode ter nenhum dos caracteres `$,.:;@`



Tags (2)

- Para criar uma tag para todo o módulo:

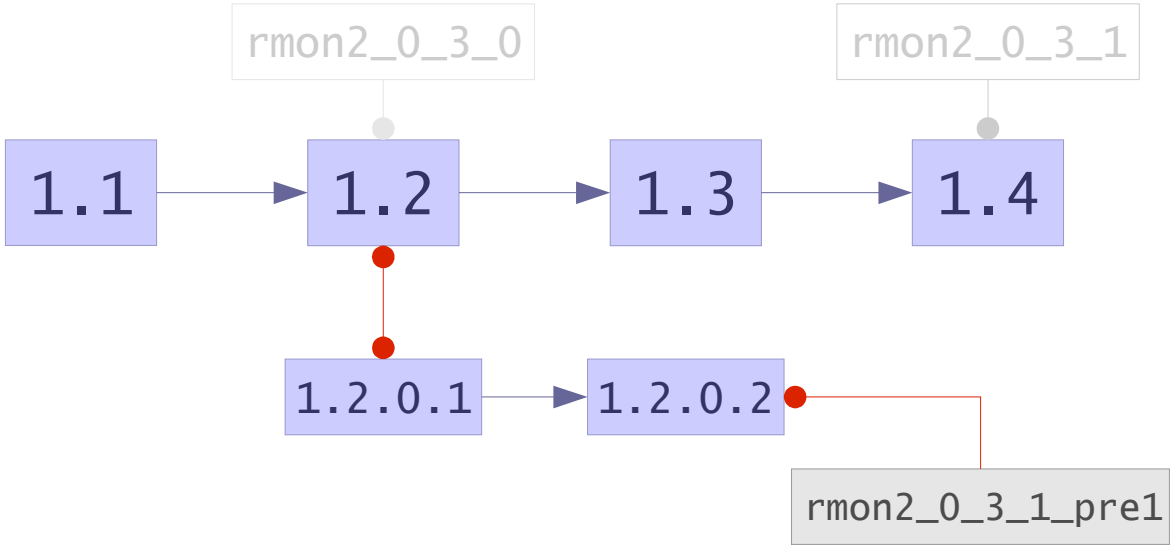
```
$ cd ~/sandbox/rmon2
$ cvs tag rmon2_0_3_0
```

- É possível criar tags para determinados arquivos ou diretórios:

```
$ cd ~/sandbox/rmon2
$ cvs tag rmon2_0_3_0_src  src/
```

Ramificações – Branches

- Servem para criar uma subdivisão no projeto, que possui um único tronco inicialmente (HEAD), utilizando marcas (tags) para determinar a última revisão da ramificação
- É possível trabalhar nesse novo ramo sem perturbar o tronco principal, o que nem sempre é verdade com tags



Criando Ramificações

- Para criar uma ramificação de um módulo retirado, usa-se o parâmetro `-b` (de branch) no comando CVS `tag`:

```
$ cd ~/sandbox/rmon2
$ cvs tag -b rmon2_0_3_1_pre1
cvs tag: Tagging .
T COPYING
T Doxyfile
T ERRORS
T INSTALL
T Makefile
...
T src/trasserlib.h
```

Acessando Marcas (Tags)

- É possível fazer a retirada de um módulo (checkout) solicitando diretamente a marca com o parâmetro `-r`:

```
$ cvs checkout -r rmon2_0_3_0_pre7 rmon2
```

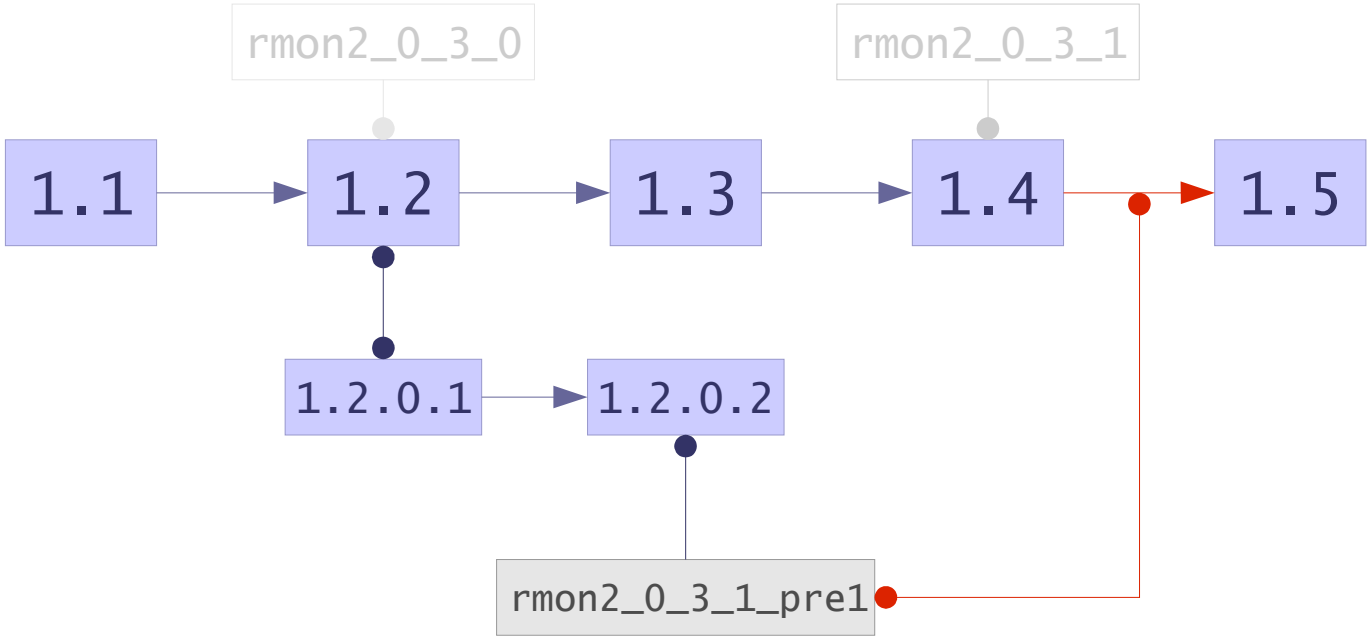
- Caso o módulo já tenha sido retirado, é possível atualizar a área de trabalho sincronizando com a marca:

```
$ cd ~/sandbox/rmon2
$ cvs update -r rmon2_0_3_0_pre7
```

- A marca é um parâmetro persistente (sticky), ou seja, não é necessário informar para o CVS a marca que se deseja nos comandos CVS que forem executados posteriormente
- A menos, é claro, que a intenção seja trocar de marca

Unindo Ramificações (1)

- Em um dado momento, pode-se desejar que um determinado ramo seja incorporado novamente ao tronco principal



Unindo Ramificações (2)

- Existem várias formas de se fazer isso
- Uma forma razoavelmente segura de fazer isso é utilizar uma outra área de trabalho para buscar o tronco, e então incorporar o ramo com o parâmetro -j (join):

```
$ cd /tmp
$ cvs checkout -j rmon2_0_3_0_pre7 rmon2
```

- Se o CVS indicar que a união foi com sucesso (isto é, sem conflitos), envia-se os resultados para o repositório:

```
$ cd rmon2
$ cvs commit
```

Solucionando Conflitos

- Quando 2 ou mais usuários modificarem a mesma seção de um arquivo de forma que o CVS não consiga mesclá-las automaticamente, ocorre um conflito
- Os conflitos devem ser solucionados manualmente, editando-se o arquivo com conflito:

```
<<<<<<< rmon2_main.c
#include <stdio.h>
=======
#include <stdlib.h>
>>>>>>> 0.4.2.2
```

- O usuário deve solucionar os conflitos, remover as marcas incluídas pelo CVS e então enviar ao repositório (commit)

Exportando um Módulo

- Quando se retira um módulo do repositório (checkout), o CVS inclui vários arquivos de controle, que não são úteis para usuários que não pretendem alterar o módulo
- Para isso, usa-se o comando export para gerar uma cópia do módulo sem os arquivos de controle, indicando uma marca de referência (tag), pronto para distribuição:

```
$ cd /tmp
$ cvs export -r rmon2_0_3_0_pre7 rmon2
cvs export: Updating rmon2
U rmon2/COPYING
U rmon2/Doxyfile
...
$ tar cfz rmon2-0.3.0-pre7.tar.gz rmon2
```

Patches (1)

- Patches são arquivos especiais que, ao invés de conter todo o conteúdo de uma nova versão (por exemplo), contêm apenas as diferenças de uma versão anterior (geralmente a penúltima) para a mais atual (última versão)
- Eles também tem suas esquisitices: podem ser minúsculos, contendo algumas poucas linhas de texto; ou gigantes, podendo ter o dobro do tamanho do arquivo original, no caso de uma versão ser totalmente diferente da outra
- Patch é o formato padrão para contribuição em projetos de software livre, e podem ser gerados facilmente pelo CVS

Patches (2)

- Exemplo:

```
$ cvs diff Makefile
Index: Makefile
===================================================================
RCS file: /home/rnsanchez/CVS/rmon2/Makefile,v
retrieving revision 1.1.1.1
diff -r1.1.1.1 Makefile
9c9
< VERSAO                = 0.3.0-pre6
---
> VERSAO                = 0.3.0-pre7
```

Patches (3)

- O formato unificado é o preferido, bastando adicionar o parâmetro -u ao comando
- O comando rdiff é similar ao diff, com a vantagem de não precisar realizar uma retirada (checkout) antes:

```
$ cvs rdiff -u -r rmon2_0_3_0_pre6 -r rmon2_0_3_0_pre7 rmon2 >
  ~/sandbox/rmon2-0.3.0-pre6-pre7.patch
cvs rdiff: Diffing rmon2
cvs rdiff: Diffing rmon2/doc
...
```

- Aplica-se o patch usando:

```
$ cd ~/sandbox/rmon2-0.3.0-pre6
$ patch -p1 < ~/sandbox/rmon2-0.3.0-pre6-pre7.patch
```

Abreviações

- No arquivo ~/.cvsrc, pode-se incluir atalhos para os comandos, para que não seja necessário digitar parâmetros que o usuário sempre use
- Exemplo:

```
cvz -z6
diff -u
status -v
```

- Isso fará com que:
  - Todos os comandos CVS usem compressão (-z6)
  - O comando cvs diff gere sempre diff unificado (-u)
  - O comando cvs status sempre seja detalhado (-v)

Chega!

- Referências recomendadas:
  - Caetano, Cristiano. CVS: Controle de Versões e Desenvolvimento Colaborativo. ISBN 8575220519
  - Neubert, Marden. CVS: Guia de Consulta Rápida. ISBN 857522056X
  - CVS Quick Reference Card. `http://refcards.com/refcards/cvs/index.html`
  - CVS Quick Reference Card. `http://tnerual.eriogerg.free.fr/cvs.html`
  - CVS Home. `http://www.cvshome.org`