

# Introdução ao CVS

Rui Carlos A. Gonçalves

29 de Agosto de 2008

## Resumo

Neste documento pretende-se descrever de forma sucinta o que é um sistema de controlo de versões, qual a sua utilidade e ensinar o passos básicos para usar o **CVS**, uma das aplicações mais usadas para este fim.

## Conteúdo

<b>1</b>	<b>Introdução</b>	<b>2</b>
<b>2</b>	<b>O cvs</b>	<b>2</b>
2.1	Criar um repositório . . . . .	2
2.2	Importar um projecto . . . . .	2
2.3	Baixar um projecto . . . . .	3
2.4	Devolver ao repositório os arquivos alterados . . . . .	3
2.5	Trabalhando em grupo . . . . .	4
2.6	Versões . . . . .	4
2.7	Visualizar informações do projecto . . . . .	5
<b>3</b>	<b>Conclusão</b>	<b>5</b>

# 1 Introdução

Um sistema de controlo de versões é uma aplicação que permite gerir várias versões de ficheiros. Este tipo de aplicação é particularmente útil no desenvolvimento de *software*. As principais vantagens que este nos trás são:

- facilitar o acesso às várias versões pela qual um determinado projecto passou, permitindo analisar facilmente a sua evolução;
- possibilitar que várias pessoas trabalhem num mesmo ficheiro ao mesmo tempo, sendo fácil *juntar* as várias versões;
- permitir controlar facilmente as versões estáveis, sendo fácil recuperá-las caso haja algum problema.

## 2 O cvs

O **CVS** (Concurrent Version System) é um dos sistemas de controlo de versões mais usados actualmente. Trata-se de um *software* de código aberto e pode ser baixado gratuitamente (mais informações podem ser encontradas no *site* <http://ximbiot.com/cvs/wiki>).

Nas próximas secções serão indicados os passos necessários para se começar a usar este programa em sistemas *Unix*.

Informações mais detalhadas sobre os comandos que serão indicado podem ser obtidas através do *help* (`cvs -H <comando>`) ou através das *man pages* (`man cvs`) do **CVS**.

### 2.1 Criar um repositório

O primeiro passo quando pensamos em usar o **CVS** consiste em criar um repositório<sup>1</sup>. Para tal devemos definir a variável `CVSROOT`. Por exemplo, para definir o directório `/home/xxx/cvs` como repositório, usaríamos os seguintes comandos:

```
CVSROOT=/home/xxx/cvs
export CVSROOT
```

Para não termos que executar este comando em todas a sessões, podemos colocar os comandos no ficheiro `/etc/profile` ou `~/.bashrc`.

De seguida executamos o comando `cvs init` que coloca no repositório uma pasta com arquivos de administração do **CVS**.

Agora é só importar os projecto para o repositório.

### 2.2 Importar um projecto

Quando trabalhamos com o **CVS** não é muito aconselhável renomear/mover ficheiros ou pastas, pois pode criar *confusões* para o **CVS**. Como tal, a primeira coisa a fazer quando iniciamos um projecto, é criar a estrutura de directórios.

De seguida colocamos então os ficheiros no repositório. Para tal usamos o comando:

---

<sup>1</sup>um repositório é um directório onde são armazenados todos os ficheiro de um projecto, assim como ficheiros criados pelo programa que permitem controlar as versões dos outros.

```
cv$ import -m "mensagem inicial" dir xxxx start
```

onde `dir` é a pasta dentro do repositório onde o projecto será colocado, `xxxx` é a *vendor tag* e `start` é a *release tag*. Isto pode não fazer muito sentido em alguns contextos, mas o **CVS** obriga a que esteja presentes. Caso não seja especificada nenhuma mensagem (o que no exemplo foi feito através da opção `-m`) será aberto um editor para que a mensagem seja introduzida. Depois disto, todos os ficheiros/pastas contidos no directório actual foram colocados no repositório em `CVSROOT/dir`.

Caso o projecto já estivesse iniciado o processo seria semelhante: seleccionava-se o directório do projecto (`cd <directorio-projecto>`) e depois executava-se o comando indicado atrás.

### 2.3 Baixar um projecto

Sempre que queremos alterar o projecto é necessário baixá-lo para um local de trabalho (que pode ser a nossa *home*, uma pasta temporária, etc). Para tal usamos o seguinte comando:

```
cv$ checkout proj
```

onde `proj` é directório do projecto que queremos baixar. Este comando criou uma nova pasta no directório actual com o nome `proj` que contém todos os ficheiros do projecto, bem como uma pasta (denominada **CVS**) que possui arquivos para uso interno do **CVS**.

### 2.4 Devolver ao repositório os arquivos alterados

Depois de efectuadas as alterações aos ficheiros é necessário actualizar o repositório. Para tal usamos o comando:

```
cv$ commit proj
```

Mais uma vez `proj` é o nome do projecto em que estamos a trabalhar. O **CVS** abrirá um editor de texto (normalmente o *Vim*, mas podemos escolher outro através da variável `CVSEEDITOR`) onde devemos descrever as alterações feitas aos vários ficheiros. Para evitar a abertura do editor de texto, podemos indicar uma mensagem junto com o comando `commit` (tal como no `import`, isto pode ser feito através da opção `-m`).

Neste caso ficaria:

```
cv$ commit -m "mensagem ..." proj
```

Depois de actualizado o repositório, podemos eliminar a cópia do projecto criada. Por exemplo, fazendo:

```
rm -r proj
```

A forma mais correcta de o fazer é, no entanto, através dos comandos:

```
cv$ release -d proj
```

Desta forma verificamos se todas as alterações foram *guardadas* no repositório. A opção `-d` elimina a copia local.

## 2.5 Trabalhando em grupo

Uma das vantagens de se usar o **CVS** é tornar mais simples o trabalho em grupo. Por exemplo, suponhamos que duas pessoas baixaram o projecto ao mesmo tempo e efectuaram algumas alterações. Caso queiramos obter as últimas versões dos ficheiros (incluindo as alterações efectuadas pelo outro utilizador, que entretanto já actualizou o repositório) podemos usar o comando **update**, que baixa novamente os ficheiros modificados (e apenas estes).

Os dois utilizadores podem até efectuar alterações no mesmo ficheiro. Quando confirmarem as alterações o **CVS** tentará conciliá-las. Nos casos em que isso não for possível, será solicitada a intervenção do utilizador.

## 2.6 Versões

O **CVS** atribui a cada arquivo um número de versão, que pode variar de ficheiro para ficheiro. Normalmente a primeira versão recebe o número 1.1, sendo que este é alterado sempre que alteramos um ficheiro (1.1 → 1.2 → 1.3 → ...).

Tendo em conta que nem todos os ficheiros de um projecto têm a mesma versão num determinado momento, pode ser útil criar um *identificador* que fique associado às versões de cada ficheiro numa determinada data. Para isso só temos que criar uma *tag*<sup>2</sup>. O comando a usar é:

```
cv$ tag v2
```

sendo *v2* o nome da *tag* criada.

Depois disto sempre que quisermos baixar esta versão do projecto basta fazer:

```
cv$ checkout -r v2 proj
```

Também é possível criar uma *tag* apenas para um ficheiro. Para tal é só usar o comando referido anteriormente, indicando o nome do ficheiro ao qual queremos associar a *tag*.

É possível eliminar, mover e mudar o nome de uma *tag*. Vejamos alguns exemplos:

```
cv$ tag -d v2
cv$ tag -r v2_1 v3_1 file.txt
cv$ tag -d v2_1 file.txt
cv$ tag -r 3.1 -F v3_1 file.txt
```

Esta sequência de comando começa por apagar a *tag v2* (associada a todos os ficheiros do projecto), depois os dois comandos seguintes alteram o nome da *tag v2.1* para *v3.1* (primeiro criando uma cópia da *tag* anterior, eliminando de seguida a *tag* antiga), *tag* esta que está associada ao ficheiro *file.txt*. Por último, indicamos que a *tag v3.1* (do ficheiro *file.txt*) vai passar a identificar a versão 3.1.

Alternativamente, podíamos ter *unificado* o número de versão de todos os arquivos do projecto. Por exemplo, para que todos os arquivos ficassem com a versão 4.0 faríamos:

```
cv$ commit -r 4.0
```

De referir que todos os arquivos deviam ter um número de versão inferior a 4.0, caso contrário a operação não será bem sucedida.

---

<sup>2</sup>uma *tag* é uma espécie de *fotografia* do repositório num determinado momento.

## 2.7 Visualizar informações do projecto

O `CVS` disponibiliza comandos que permitem consultar as versões/*tag*'s de um arquivo, bem como ver as alterações efectuadas e as mensagens a elas associadas.

```
cv$ log
cv$ log file.txt
cv$ status file.txt
```

O primeiro comando mostra as mensagens associadas às várias alterações feitas nos vários ficheiros do projecto. Já o segundo apenas mostra as mensagens relativas ao ficheiro `file.txt`. O último mostra algumas informações a respeito do arquivo `file.txt`.

## 3 Conclusão

Ao longo deste texto foram indicadas algumas das potencialidades do uso de *software* de controlo de versões, bem como informações básicas sobre o uso de uma ferramenta deste tipo. Ficaram ainda por referir alguns aspectos mais *avançados*, como a criação de ramificações ou a configuração de um servidor. Tais aspectos deverão ser abordados em futuras versões. Pensa-se, no entanto, que foram indicados aspectos suficientes para justificar a sua utilização por parte das pessoas que trabalham em projectos de *software*. Sublinhava, por fim, que este texto foi redigido para uso próprio e, como tal, não pretende ser nenhuma referência sobre o tema e poderá até ter algumas incorrecções.