Sistemas de Controlo de Versões

Laboratórios de Informática I

MIEI

O desenvolvimento de *software* é cada vez mais complexo, e obriga a que uma equipa de programadores possa desenvolver uma mesma aplicação ao mesmo tempo, sem se preocuparem com os detalhes do que outros membros dessa mesma equipa estejam a fazer. É evidente que alterações concorrentes (realizadas por diferentes pessoas ao mesmo tempo) podem provocar conflitos quando várias pessoas editam o mesmo bocado de código.

Além disso, não nos devemos esquecer que algumas alterações a um programa, no sentido de corrigir ou introduzir alguma funcionalidade, podem elas mesmas conter erros, e pode por isso ser necessário repor uma versão prévia da aplicação, anterior a essa alteração.

Para colmatar estes problemas são usados sistemas de controlo de versões.

1 Panorama nos Sistemas de Controlo de Versões

Existe um grande conjunto de sistemas que permitem o desenvolvimento cooperativo de *software*. Todos eles apresentam diferentes funcionalidades mas os seus principais objetivos são exatamente os mesmos.

Habitualmente divide-se este conjunto em dois, um conjunto de sistemas denominados de *centralizados*, e um outro de sistemas *distribuídos*:

- Sistemas de controlo de versões centralizados:
  - Concurrent Versions System (CVS): `http://www.nongnu.org/cvs/`;
  - Subversion (SVN): `https://subversion.apache.org/`;
- Sistemas de controlo de versões distribuídos:
  - Git: `http://git-scm.com/`;
  - Mercurial (hg): `http://mercurial.selenic.com/`;
  - Bazaar (bzr): `http://bazaar.canonical.com/en/`;

Estes são apenas alguns exemplos dos mais usados. A grande diferença entre os centralizados e os distribuídos é que, nos centralizados existe um repositório, denominado de servidor, que armazena, a todo o momento, a versão mais recente do código fonte. Por sua vez, nos distribuídos, cada utilizador tem a sua própria cópia do repositório, que podem divergir, havendo posteriormente métodos para juntar repositórios distintos.

2 Instalação do Subversion

Na disciplina de Laboratórios de Informática I será utilizado o sistema *Subversion* que será, de agora em diante, designado apenas por *svn* (nome do comando usado no terminal).

O sistema operativo Mac OS X já inclui uma versão do *svn* instalada. Para Linux, depende da distribuição de Linux em causa. Num *Ubuntu* poderá ser instalado com

```
sudo apt-get install subversion
```

Do mesmo modo, em *Fedora* poderá ser instalado com

```
sudo yum install subversion
```

Em sistemas operativos Windows sugere-se o uso da aplicação TortoiseSVN[1].

3 Repositório e Utilizadores

O repositório centralizado que será usado na disciplina de LI I está hospedado num servidor do Departamento de Informática: `svn://svn.alunos.di.uminho.pt`. Cada grupo tem um repositório privado, e cada elemento do grupo tem um nome de utilizador (*username*) e uma palavra-chave (*password*) diferentes.

Os nomes de utilizador são constituídos por `li1` seguido pelo número do grupo, como por exemplo, `g999`, seguido do número do utilizador, por exemplo, `a1`. Nos exemplos que se seguem será usado o utilizador `2017li1g999a1`, que deverá ser substituído pelo respectivo nome do utilizador.

4 Uso do Subversion

4.1 Checkout

A primeira etapa no uso do *svn* corresponde ao processo de ir buscar ao servidor a versão mais recente do repositório. **Este processo só deverá ser executado uma vez**.

O comando para tal é `svn checkout` seguido do endereço do repositório (tenha em atenção que o número do grupo deve ser devidamente alterado) e do nome do utilizador que irá usar esta cópia do repositório. Por exemplo, o utilizador *a1* do grupo *g999* deverá executar o seguinte comando.

```
$ svn checkout svn://svn.alunos.di.uminho.pt/2017li1g999 --username 2017li1g999a1
```

Depois de inserir a respectiva *password* correctamente, a seguinte mensagem deverá aparecer.

```
Checked out revision 1.
```

O nome *revision* é dado a um inteiro que identifica a versão atual do código do repositório. Sempre que algum utilizador atualiza o código no repositório este valor será incrementado.

O resultado da execução deste comando é uma pasta, criada na pasta atual, com o nome do repositório (`2017li1g999` no exemplo acima). Esta pasta será a raiz do repositório. O repositório de cada grupo foi previamente preenchido com pastas e ficheiros, sendo que após o primeiro *checkout* contém a seguinte estrutura:

```
.
|---- src/
      |---- LI11718.hs
      |---- Tarefa1_2017li1g999.hs
      |---- Tarefa2_2017li1g999.hs
      |---- Tarefa3_2017li1g999.hs
```

Esta estrutura de ficheiros do repositório é **obrigatória**. Na pasta `src` deve estar o código fonte Haskell, devendo as três tarefas da primeira fase ser implementadas nos três respectivos ficheiros lá colocados.

[1] `http://tortoisesvn.net/`

4.2 Adição de Novas Pastas e Ficheiros

O passo seguinte corresponde a adicionar novos ficheiros ou pastas que queiramos armazenar no repositório. Como exemplo, vamos adicionar um ficheiro `README.txt` ao repositório central. Crie o ficherio `README.txt` na directoria do repositório, com uma breve descrição do projecto, o número e o nome de cada elemento do grupo.

Para adicionar o ficheiro no repositório, realiza-se a seguinte operação a partir da raiz do repositório:

```
$ svn add README.txt
```

ao que o *svn* deverá responder:

```
A         README.txt
```

Isto indica que o ficheiro ficou marcado para ser seguido pelo *svn* (daí o `A` na primeira coluna). No entanto, o ficheiro ainda não foi enviado para o servidor. Apenas o marcamos como um ficheiro que mais tarde deverá ser enviado para o servidor.

4.3 Status

Um comando extremamente simples, mas bastante útil, designado por *status*, permite ver o estado atual do repositório local (sem realizar qualquer ligação ao servidor).

Depois de termos adicionado o ficheiro `README.txt`, ao executar o comando `svn status` obtemos:

```
$ svn status
A       README.txt
```

Isto indica que neste momento o repositório local tem um ficheiro marcado como adicionado (marca `A`).

Se criar mais um ficheiro de teste, denominado `exemplo.txt`, mas não o adicionar, ao executar o comando *status* obtém-se:

```
$ svn status
A       README.txt
?       exemplo.txt
```

O ponto de interrogação indica que o *svn* não sabe nada sobre aquele ficheiro, e que portanto o irá ignorar em qualquer comando executado.

Para que este ficheiro seja adicionado ao repositório terá que usar o comando `svn add exemplo.txt`, tal como usado anteriormente.

```
$ svn add exemplo.txt
$ svn status
A       README.txt
A       exemplo.txt
```

4.4 Commit

O comando de adição de ficheiros **só é necessário para ficheiros ou pastas novos**. Quando são realizadas alterações a ficheiros previamente adicionados não é necessário realizar esse comando.

Para enviar as alterações e os novos ficheiros ou pastas para o servidor central é necessário realizar um processo designado por *commit*. Isto poderá ser feito através do comando `svn commit`.

```
$ svn commit -m "Modificado o README e criados novos ficheiros de teste"
Adding        README.txt
Adding        exemplo.txt
Transmitting file data .
Committed revision 2.
```

No comando *commit* executado foi adicionada uma opção (`-m`) que é usada para incluir uma mensagem explicativa das alterações que foram introduzidas ao repositório. É possível não especificar a opção, e nesse caso o *svn* irá abrir um editor de texto que tenha sido configurado, para que o utilizador o use para adicionar a mensagem. É boa prática adicionar uma mensagem clara em cada *commit*.

As linhas seguintes indicam que os ficheiros foram adicionados e enviados para o servidor. Finalmente, é indicada a revisão em que a cópia local do repositório ficou.

4.5 Update

Sendo que o *svn* é especialmente útil no desenvolvimento cooperativo, vamos ver o que acontece quando um outro utilizador altera o repositório. Suponhamos então que um outro utilizador alterou o ficheiro `README.txt`.

Um utilizador de *svn* deve atualizar a sua cópia local do repositório sempre que possível, para que quaisquer alterações que tenham sido incluídas por outros programadores sejam propagadas do repositório central para a sua cópia local.

Este processo é feito através do comando `svn update`. Por exemplo:

```
$ svn update
Updating '.':
U    README.txt
Updated to revision 3.
```

Ao realizar a atualização o *svn* indica que o ficheiro `README.txt` foi atualizado (apresentando a marca `U`, de *Updated*), e indica que a revisão atual do repositório é a 3.

Nem sempre os programadores estão a trabalhar em ficheiros distintos. Supondo que dois utilizadores alteraram o mesmo ficheiro em simultaneo, mas em zonas diferentes do ficheiro (por exemplo, cada um modificou apenas o seu nome no ficheiro `README.txt`), e que o primeiro já fez *commit* da sua alteração, então o segundo irá obter o seguinte resultado ao atualizar a sua cópia:

```
$ svn update
Updating '.':
G    README.txt
Updated to revision 4.
```

A marca `G` antes do nome do ficheiro indica que houve uma junção da versão que estava no repositório com a nossa versão local (*merGe*). No entanto, o *svn* foi capaz de lidar com a alteração concorrente sem problemas, e portanto, poderá ser feito o *commit* destas últimas alterações:

```
$ svn commit -m "Modificado nome de elemento do grupo"
Sending        README.txt
Transmitting file data .
Committed revision 5.
```

Em algumas situações poderá acontecer que dois utilizadores tenham editado a mesma zona do ficheiro, e portanto, que o *svn* não tenha conseguido juntar as duas versões. Nesta situação, ao realizar o *update* surgirá:

```
$ svn update
Updating '.':
Conflict discovered in '/Users/ambs/tmp/2017li1g999/README.txt'.
Select: (p) postpone, (df) diff-full, (e) edit,
        (mc) mine-conflict, (tc) theirs-conflict,
        (s) show all options:
```

Neste caso o *svn* diz que encontrou um conflito no ficheiro `README.txt` e permite escolher uma de várias opções, sendo as mais usadas são:

(p) postpone esta opção indica que o *svn* deverá manter o ficheiro com conflito. Nesta situação o utilizador terá de verificar manualmente o ficheiro para resolver o conflito.

(e) edit com esta opção o *svn* abre o ficheiro com conflito num editor, permitindo que o utilizador o corrija de imediato.

(mc) mine-conflict esta opção, que deve ser usada com cuidado, descarta a versão que está no repositório central e considera a versão local como a correta.

(tc) theirs-conflict esta opção é usada quando sabemos que a nossa alteração é irrelevante, e portanto, pretendemos ficar apenas com a versão do repositório central.

Ao editar um ficheiro com conflito aparecerão marcas deste género:

```
<<<<<<< .mine
Repositorio de Testes
=======
repositório de testes
>>>>>>> .r5
```

Isto indica a zona com conflito. A parte superior corresponde ao texto produzido pelo utilizador atual (texto na versão local do repositório). A parte inferior corresponde à versão que está no repositório central. Nesta situação é nosso dever remover as marcas (as linhas com `====`, `<<<<` e `>>>>`) e optar por uma das partes (ou então, criar uma nova que resolva o conflito).

Depois de resolver um conflito o utilizador deverá:

- indicar que o conflito foi resolvido:

```
$ svn resolved README.txt
Resolved conflicted state of 'README.txt'
```

- realizar nova atualização da cópia local (e resolver novos conflitos que possam surgir):

```
$ svn update
Updating '.':
At revision 6.
```

- estando tudo correto, então poder-se-á fazer o *commit*:

```
$ svn commit -m "resolvido o conflito"
Sending        README.txt
Transmitting file data .
Committed revision 7.
```

4.6 Remove

Vamos agora ver como remover ficheiros do repositório. Isto poderá ser feito através do comando `svn remove`, seguido do nome do ficheiro. Para remover o ficheiro `exemplo.txt` faz-se:

```
$ svn remove exemplo.txt
D         exemplo.txt
```

Tal como na operação *add*, temos que fazer *commit* para que um ficheiro marcado para ser apagado (marca `D` de *Delete*) seja efetivamente apagado no repositório central.

```
$ svn commit -m "Remove file exemplo"
Deleting       exemplo.txt
Committing transaction...
Committed revision 8.
```

Note que embora o ficheiro tenha sido removido da directoria de trabalho, ele ficará guardado no servidor. Portanto se tal for necessário é possível reaver o ficheiro. Se tal for necessário, procure informação sobre o comando `svn revert` em [1].

4.7 Outros comandos úteis

Segue uma lista de alguns comandos

- `svn help COMMAND`
- `svn copy SOURCE TARGET`
- `svn move SOURCE TARGET`
- `svn log`

Note que para que o `svn log` apresente a informação actualizada deverá sempre fazer `svn update` antes.

Referências

Sugere-se a consulta de documentação do *svn*, nomeadamente:

[1] O livro "*Version Control with Subversion*" da O'Reilly, disponível em formato eletrónico em `http://svnbook.red-bean.com/`;

[2] Os comandos disponíveis no *svn*, usando o comando `svn help`, ou a documentação específica de um comando, com `svn help` *comando* em que *comando* é substituído por um dos comandos do *svn*.