

Introdução ao Subversion

Thiago Camargo Fernandes

10 de Outubro de 2007

Conteúdo

1	Prefácio	2
1.1	O que é o Subversion?	2
1.2	A arquitetura do Subversion	2
1.3	Componentes do Subversion	3
2	Conceitos Fundamentais	5
2.1	O Modelo Cópia-Modifica-Funde	5
2.2	O Modelo Trava-Modifica-Destrava	7
2.3	Cópias de Trabalho	7
2.4	Revisões	7
3	Uso Básico	9
3.1	Help	9
3.2	Colocando dados em seu Repositório	9
3.2.1	svn import	9
3.2.2	Checkout	10
3.3	Ciclo Básico de Trabalho	11
3.3.1	Atualizando sua Cópia de Trabalho	11
3.3.2	Fazendo suas mudanças	12
3.3.3	Examinando suas Mudanças	12
3.3.4	Visão Detalhada das Mudanças	14
3.4	Desfazendo Mudanças	14
3.5	Resolvendo Conflitos	14
3.5.1	Enviando suas Mudanças	15
3.6	Examinando Histórico	15

Capítulo 1

Prefácio

1.1 O que é o Subversion?

O Subversion é um aplicativo de código aberto para controle de versões. Ele tem como finalidade gerenciar arquivos e diretórios, e todas as mudanças feitas neles no decorrer do tempo. Dessa forma, ele permite a recuperação de versões antigas dos dados e também o exame do histórico das mudanças feitas.

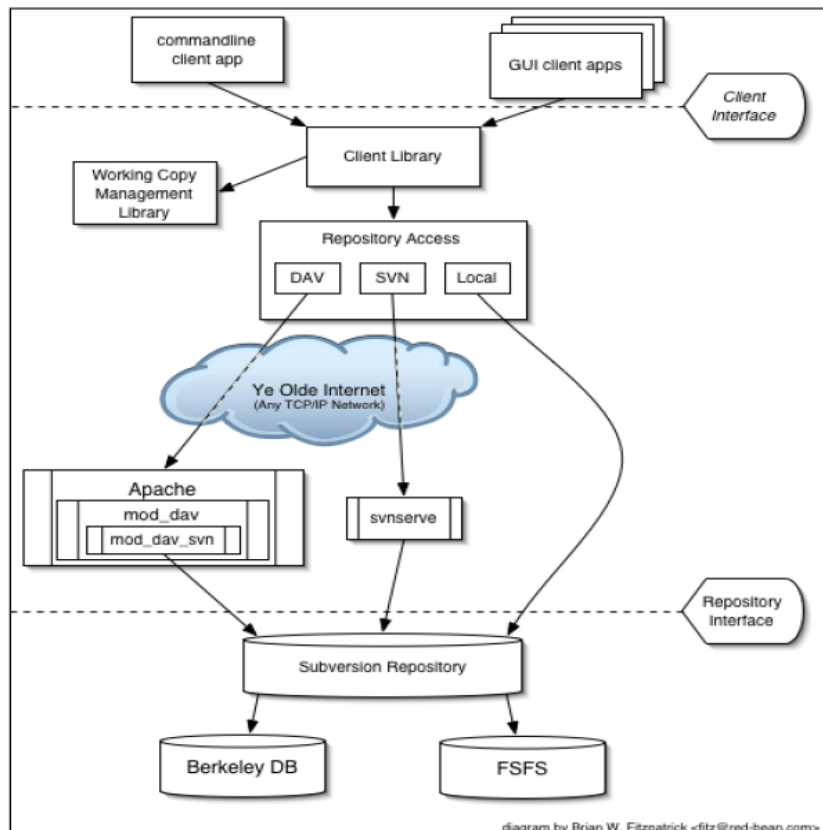
O Subversion pode operar através de redes, o que permite ele ser usado por pessoas em diferentes computadores e em qualquer lugar do mundo através da internet.

Nos casos de várias pessoas trabalhando num mesmo conjunto de dados, o Subversion oferece mecanismos de tratamento de colisões, ou seja, ele trata ou pelo menos alerta, que dados podem ser perdidos ou sobrescritos, por conta de duas ou mais pessoas estarem alterando o mesmo dado ao mesmo momento.

Deste modo, o Subversion é uma ferramenta que facilita o desenvolvimento de projetos, pois funciona como uma máquina do tempo para recuperações de versões antigas, e também proporciona a possibilidade de várias pessoas trabalharem com os mesmos dados sem que ocorram maiores problemas.

1.2 A arquitetura do Subversion

De um lado temos o repositório que armazena as versões de todos os projetos. Do outro lado temos o programa cliente do Subversion que gerencia as cópias locais de uma ou mais versões dos projetos. Entre esses dois extremos existem múltiplas rotas através de várias camadas para o acesso ao repositório, como podemos verificar na figura abaixo.



1.3 Componentes do Subversion

Depois de instalado, o Subversion possui alguns arquivos executáveis:

- **svn**: A linha de comando do programa cliente;
- **svnversion**: Reporta o estado de uma cópia local, em termos da revisão dos itens presentes;
- **svnlook**: Uma ferramenta para inspeção direta do repositório do Subversion;
- **svnadmin**: Uma ferramenta para a criação ou reparo de um repositório do Subversion;
- **svndumpfilter**: Um programa que filtra partes inúteis do repositório do Subversion;
- **mod_dav_svn**: Plug-in para o servidor HTTP Apache, para conexão por rede;
- **svnserve**: O programa servidor padrão, que pode ser executável como daemon ou chamado por SSH;

- `svnsync`: Um programa para espelhar incrementalmente um repositório a outro em rede.

Capítulo 2

Conceitos Fundamentais

2.1 O Modelo Cópia-Modifica-Funde

Quando duas ou mais pessoas estão utilizando simultaneamente um mesmo conjunto de dados, já sabemos que podem haver colisões. O modelo copia-modifica-funde é uma das soluções encontradas para se evitar esse tipo de colisão. Este modelo consiste no programa cliente de cada usuário criar uma cópia de trabalho local da versão que o usuário deseja realizar a alteração. Dessa maneira, os usuários podem trabalhar simultaneamente e independentemente, modificando suas próprias cópias, e só depois as cópias são fundidas numa versão final.

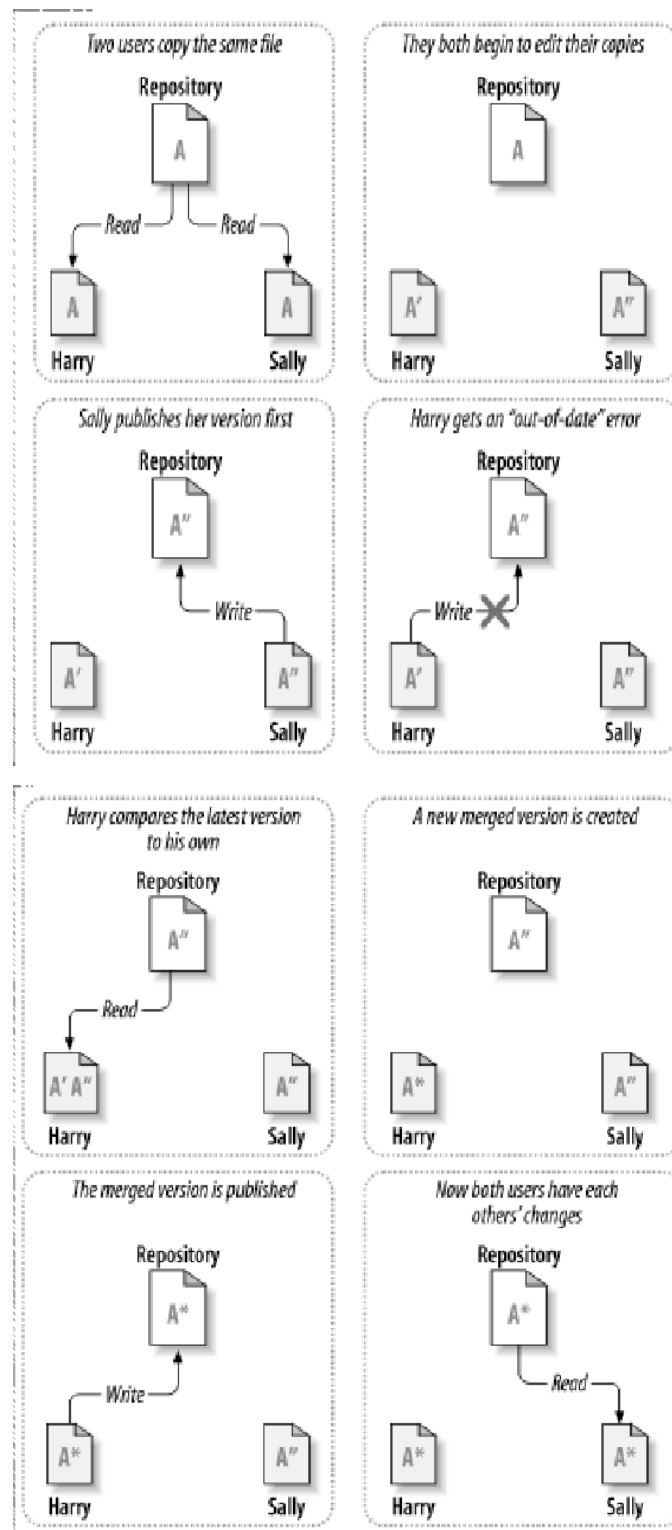
O Subversion além de utilizar esse modelo, ele auxilia na fusão dos dados. Não podemos dizer que ele faz a fusão pois, apenas em uma parte dos casos ele consegue resolver esse problema, nos casos restantes, ele depende da intervenção humana pra julgar o que deve ser feito.

Para um melhor entendimento de como o Subversion funciona temos o seguinte exemplo:

João e Maria criam, cada um, uma cópia de um mesmo projeto em seus computadores. Ambos trabalham concorrentemente e fazem alterações num mesmo arquivo X. Maria envia suas alterações ao repositório como uma nova versão. No momento em que João vai enviar suas alterações, o repositório o avisa que seu arquivo X está desatualizado. Então João pede para que o programa cliente faça a fusão do arquivo X local com o arquivo X do repositório. Se as alterações de Maria não coincidirem com as alterações de João, então o Subversion faz normalmente a fusão e João pode enviar essa nova versão ao repositório.

Caso as mudanças que Maria fez coincidirem com as de João, o programa cliente avisará que há um conflito entre os arquivos X, e então, será João que deverá resolver isso manualmente, avisar o Subversion que o conflito foi resolvido, para então, poder enviar o arquivo para o repositório, criando uma nova versão do projeto.

Os diagramas abaixo podem ilustrar melhor como esse modelo funciona.



Tudo isso funciona muito bem quando temos arquivos que podem ser fundidos,

como os arquivos de texto. Mas o que acontece com um arquivo binário? Se duas pessoas fizessem mudanças ao mesmo tempo em uma imagem, então, uma delas acabaria perdendo o seu trabalho. Isso é o que um aplicativo de controle de versão se propõe a evitar. A solução para isso é o modelo trava-modifica-destrava, que veremos a seguir.

2.2 O Modelo Trava-Modifica-Destrava

O Subversion adota esta solução para tratar os arquivos binários, já que o modelo copia-modifica-funde não funciona neste caso.

Este modelo funciona de uma forma muito simples. Toda vez que um usuário vai fazer alterações em um arquivo do repositório, o arquivo permanece travado, permitindo apenas que aquele usuário modifique-o e impedindo qualquer outra pessoa de alterá-lo.

Se tratando de arquivos que não podem se fundir, esse modelo tem um grande problema. Caso um usuário trave o arquivo e o esqueça nesse estado, a próxima vez que alguém tentar utilizá-lo, não conseguirá e ficará de mãos atadas, imaginando que alguma outra pessoa estará utilizando o arquivo naquele momento. Até que o segundo usuário perceba que deve haver algo de errado com o arquivo, muito tempo pode ser perdido, e então ele perderá mais tempo ainda, procurando o administrador do sistema para que o problema seja resolvido.

2.3 Cópias de Trabalho

Uma cópia de trabalho do Subversion é uma árvore de diretório comum no seu sistema local, contendo uma coleção de arquivos. Você pode editar estes arquivos da maneira que quiser. O Subversion nunca irá incorporar mudanças de outras pessoas, nem tornar as suas mudanças disponíveis para outros, sem que você diga explicitamente para ele fazer isso.

O Subversion também cria e mantém numa cópia de trabalho alguns arquivos extras. Ele cria uma pasta chamada `.svn`, também conhecida como o diretório administrativo da cópia de trabalho. Os arquivos em cada diretório administrativo ajudam o Subversion a reconhecer quais arquivos contêm mudanças não publicadas, desatualizadas, etc.

2.4 Revisões

Uma operação **svn commit** publica qualquer número de arquivos e diretórios como uma única transação atômica. Na sua cópia de trabalho você pode alterar conteúdo dos arquivos, criar, deletar, renomear e copiar arquivos e diretórios, e então enviar um conjunto completo como uma transação atômica.

Cada vez que o repositório aceita um envio, é criado um novo estado da árvore do sistema de arquivos, chamado revisão. A cada revisão é designada um único

número natural, incrementado do número da revisão anterior. A revisão inicial de um repositório ,que foi acabado de criar, é zero, e consiste em nada mais que um diretório vazio.

Capítulo 3

Uso Básico

3.1 Help

Antes de continuar lendo, você deve aprender o mais importante dos comandos do Subversion: **svn help**. Um rápido **svn help SUBCOMMAND** irá descrever a sintaxe e o comportamento do subcomando.

```
$ svn help import
import: Commit an unversioned file or tree into the repository.
usage: import [PATH] URL
```

```
Recursively commit a copy of PATH to URL.
If PATH is omitted '.' is assumed.
Parent directories are created as necessary in the repository.
If PATH is a directory, the contents of the directory are added
directly under URL.
```

```
Valid options:
-q [--quiet]           : print as little as possible
-N [--non-recursive] : operate on single directory only
```

3.2 Colocando dados em seu Repositório

3.2.1 svn import

O comando **svn import** é a maneira mais rápida de copiar uma árvore de arquivos sem versão num repositório. Este comando não requer uma cópia de trabalho, e os arquivos são imediatamente enviados ao repositório. Normalmente ele é utilizado quando você já possui um projeto, e quer gerenciá-lo com o Subversion.

```
$ svnadmin create /usr/local/svn/novo_repositorio
```

```
$ svn import minha_arvore file:///usr/local/svn/novo_reposito
rio/vários/projeto/
      -m "Initial import"
Adding   minha_arvore/foo.c
Adding   minha_arvore/bar.
Adding   minha_arvore/subdir
Addind   minha_arvore/subdir/quux.c
```

(Obs.: PATH é o caminho para a árvore do arquivo de sistema do Subversion)

Lembrando que depois que a importação estiver terminada, a árvore original não é convertida para uma cópia de trabalho.

3.2.2 Checkout

Quando você faz o checkout do repositório, é criada uma cópia de trabalho na sua máquina local. Essa cópia vai conter a última revisão do projeto que você especificou na linha de comando:

```
$ svn checkout http://svn.collab.net/repos/svn/trunk
A trunk/Makefile.in
A trunk/ac-helpers
A trunk/ac-helpers/install.sh
A trunk/ac-helpers/install-sh
A trunk/build.conf
```

Checked out revision 8810.

Você também pode facilmente fazer um checkout de um subdiretório a qualquer profundidade, que esteja no repositório especificado.

```
$ svn checkout http://svn.colla.net/repos/svntrunk/subversion/
tests/cmdline/
```

No comando checkout também podemos especificar a revisão que queremos fazer uma cópia.

```
$ svn checkout -r 8810 http://svn.colla.net/repos/svntrunk/
subversion/
```

3.3 Ciclo Básico de Trabalho

O Subversion possui uma infinidade de opções e comandos, porém no dia-a-dia você normalmente vai usar apenas alguns deles.

Um ciclo de trabalho típico possui esta aparência:

1. Atualize sua cópia de trabalho
 - **svn update**
2. Faça as mudanças
 - **svn add**
 - **svn delete**
 - **svn copy**
 - **svn move**
 - **svn mkdir**
3. Examine suas mudanças
 - **svn status**
 - **svn diff**
4. Possivelmente desfça algumas mudanças
 - **svn revert**
5. Resolva os conflitos
 - **svn update**
 - **svn resolved**
6. Envie suas mudanças
 - **svn commit**

3.3.1 Atualizando sua Cópia de Trabalho

Quando você está trabalhando em um grupo com várias pessoas, você vai querer atualizar sua cópia de trabalho para receber as mudanças feitas, pelos outros desenvolvedores, desde a sua última atualização. Para isso utilize o comando **svn update**.

```
$ svn update
U foo.c
U bar.c
Updated to revision 2.
```

3.3.2 Fazendo suas mudanças

Os comando do Subversion que você usará aqui são **svn add**, **svn delete**, **svn copy**, **svn move**, e **svn mkdir**. Entretanto, se você estiver apenas editando arquivos que já existem em sua cópia de trabalho, você não precisará utilizar esses comandos. Neste caso, é o próprio Subversion que detecta as alterações nos arquivos, portanto, não precisa ser avisado sobre elas.

Abaixo temos uma breve descrição dos cinco subcomandos que são usados para fazer mudanças na árvore.

svn add foo

Marca o arquivo, diretório, ou link simbólico *foo* para ser adicionado ao repositório. Note que no caso de *foo* ser um diretório, tudo que está abaixo dele também será marcado para a adição.

svn delete foo

Marca um arquivo, diretório, ou link simbólico *foo* para ser deletado do repositório. Se *foo* for um arquivo ou link simbólico, ele é imediatamente deletado da cópia de trabalho. Caso *foo* for um diretório, ele não será deletado imediatamente da sua cópia de trabalho, apenas quando for feito o *commit*.

svn copy foo bar

Cria um novo item *bar* como uma duplicação de *foo* e automaticamente marca *bar* para a adição.

svn move foo bar

Marca *bar* para ser adicionado como uma cópia de *foo*, e *foo* é marcado para remoção.

svn mkdir blort

Cria um novo diretório chamado *blort* e o marca para adição no repositório.

3.3.3 Examinando suas Mudanças

Uma vez que acabadas as mudanças, você precisa enviá-las ao repositório, mas antes disso, normalmente é uma boa idéia dar uma olhada nas mudanças que você fez. Examinando as mudanças antes de enviá-las, você pode escrever mensagens de log mais precisas. Você pode também perceber que fez alterações que não poderiam ser feitas, e então, ter uma chance de revertê-las.

Visão Superficial das Mudanças

Para se ter uma visão superficial das mudanças, você deve usar o comando **svn status**.

Se você executar **svn status** em sua cópia de trabalho sem argumentos, ele irá detectar todas as mudanças de arquivos e de árvore que você fez.

```
$ svn status
A   stuff/loot/bloo.h
B   stuff/loot/lump.c
D   stuff/fish.c
M   bar.c
```

A primeira coluna contém códigos que mostram o status do arquivo. Esses códigos significam o seguinte:

A item O item foi marcado para adição no repositório;

C item O item está num estado de conflito. O conflito deve ser resolvido antes de ser enviado ao repositório.

D item O item foi marcado para remoção no repositório;

M item O conteúdo do item foi modificado;

Podemos também querer alguns dados a mais sobre os arquivos. Para isso, utilizamos o comando **svn status -v**.

```
$ svn status -v
A           44      23   maria   stuff/loot/bloo.c
M           44      30   joao    README
```

A primeira coluna possui os códigos de estado, assim como no comando simples. A segunda coluna mostra a revisão que se está trabalhando. A terceira e a quarta coluna mostram, respectivamente, a revisão em que o item foi modificado pela última vez, e quem o mudou.

Nenhuma das chamadas acima contatam o repositório, elas trabalham localmente comparando os metadados que estão no diretório *.svn* com a cópia de trabalho.

Para compararmos a cópia de trabalho com a do repositório, devemos utilizar a opção **-u**.

```
$ svn status -v
A   *           44      23   maria   stuff/loot/bloo.c
M           44      30   joao    README
```

Perceba que agora existe um asterisco na saída. Esse asterisco significa que aquele item da sua cópia de trabalho está desatualizado em relação ao repositório.

3.3.4 Visão Detalhada das Mudanças

Outra maneira de examinar suas mudanças é utilizando o comando `svn diff`. Dessa forma, você pode encontrar exatamente o que você modificou em cada arquivo. Ele compara o arquivo de trabalho com um arquivo que está em cache no diretório `.svn`.

A saída é dada no formato unificado do comando `diff`, que, basicamente, no início das linhas removidas insere sinais de menos, e no início das linhas adicionadas insere sinais de mais.

3.4 Desfazendo Mudanças

Após examinar as mudanças, eventualmente, pode-se encontrar alguns erros cometidos. Muitas vezes, para se consertar esses erros, acaba sendo mais fácil começar novamente. Quando você se depara com tal situação, o comando que você necessita é o `svn revert`.

Quando esse comando é utilizado, o Subversion retorna o arquivo para a versão que está guardada em cache no diretório `.svn`.

3.5 Resolvendo Conflitos

Suponha que foi executado `svn update` e o seguinte ocorreu:

```
$ svn update
U  INSTALL
G  README
C  bar.c
```

Cada sigla tem o seguinte significado:

- U** o item não tinha mudanças locais e foi atualizado com as mudanças do repositório;
- G** o item foi fundido, o que significa que, o arquivo local tinha mudanças, porém elas não se sobrepunham às mudanças do arquivo do repositório;
- C** o item está em conflito. Isso significa que as mudanças feitas no arquivo da cópia de trabalho se sobrepõem às mudanças do arquivo do repositório.

Quando arquivos são atualizados ou fundidos não é preciso se preocupar. Porém, quando um conflito ocorre, três coisas tipicamente acontecem para ajudá-lo a perceber e resolver o conflito:

- O Subversion imprime um **C** durante a atualização para lembrá-lo que o arquivo está em conflito;
- Se o Subversion considerar que o arquivo pode ser fundido, ele coloca *marcadores de conflito* no arquivo para visivelmente mostrar as áreas que se sobrepõem;

- Para cada arquivo em conflito, o Subversion coloca três arquivos extra em sua cópia de trabalho:

filename.mine Este é o arquivo como existia em sua cópia de trabalho logo antes de se fazer a atualização.

filename.rOLDREV Este é o arquivo que foi a revisão base antes de se atualizar a cópia de trabalho.

filename.rNEWREV Este é o arquivo que o programa cliente acabou de receber do servidor quando se atualizou a cópia de trabalho.

Quando um conflito ocorre, o Subversion não irá permitir que você envie a cópia daquele arquivo ao repositório até que os três arquivos sejam removidos. Desta forma, você terá três opções para resolver isso:

- Fundir o código em conflito “na mão”;
- Copiar um dos arquivos temporários encima de seu arquivo de trabalho;
- Executar **svn revert nome_arquivo** jogando fora todas as suas mudanças locais.

Uma vez resolvido o conflito, você deve executar o comando **svn resolved**. Isso irá remover os três arquivos temporários, e o Subversion não considerará mais que o arquivo está num estado de conflito.

3.5.1 Enviando suas Mudanças

Agora que você acabou de editar seus arquivos, e já fundiu todas as mudanças com as do servidor, você está pronto para enviar suas mudanças ao repositório. Para isso, você deverá utilizar o comando **svn commit**. Quando você envia uma mudança, é necessário também fazer uma mensagem de log as descrevendo.

Durante a operação de *commit*, o Subversion automaticamente irá abrir o seu editor de texto favorito para você escrever a mensagem de log. Só depois de salvar o arquivo e sair do editor, o *commit* será feito.

3.6 Examinando Histórico

Existem uma série de comandos que podem lhe dar acesso a dados do histórico do repositório, os mais utilizados são:

svn log Mostra informações como a mensagem de log, data e autor da mudança que estão juntas às revisões.

```
$ svn log
-----
r2 | maria | Mon, 15 Jul 2007 20:02:33 -0500 | 1 line
```

```
Adicionado os métodos do main()
```

```
-----  
r1 | sally | Mon, 15 Jul 2007 18:45:13 -0500 | 1 line
```

```
Importação inicial  
-----
```

Pode também ser utilizado com a opção **-r** para se especificar uma revisão:

```
$ svn log -r 5:19 # Mostra logs das revisões 5 a 19
```

```
$ svn log -r 19:5 # Mostra logs das revisões 5 a 19  
# em ordem reversa
```

```
$ svn log -r 5 # Mostra o log da revisão 5
```

svn diff Mostra detalhes ao nível das linhas de texto.

- Comparando mudanças locais

```
$ svn diff
```

- Comparando a cópia de trabalho com o repositório:

```
$ svn diff -r 3 rules.txt
```

- Comparando revisões do repositório:

```
$ svn diff -r 2:3 rules.txt
```

svn cat Imprime em sua tela o conteúdo de um arquivo de uma certa revisão.

```
$ svn cat -r 2 rules.txt
```

svn list Mostra os arquivos do diretório de uma certa revisão.

```
$ svn list http://svn.collab.net/repos/svn
```