クライアント

■ RapidSVN

http://rapidsvn.tigris.org/
設定は不要。

Tortoise を使いたくない場合は、これがいいと思う。

http://amaxi.sitemix.jp/blog/archives/1493

> **このアプリケーションの構成が正しくないため、アプリケーションを開始できませんでした。**
> **アプリケーションを再度インストールすることにより問題が解決する場合があります。**

っと表示されたら、
Microsoft Visual C++ 2005 Service Pack 1 再頒布可能パッケージ ATL のセキュリティ更新プログラム
をインストールする。

■ esvn

http://terai.xrea.jp/Subversion/eSvn.html

svn のコマンドが使える環境が必要。
http://subversion.tigris.org/
からダウンロードしてパスを通す。

・環境変数 APR_ICONV_PATH を subversion をインストールしたディレクトリの下の iconv に設定しないと文字化けする可能性あり。

■ svn + ssh

ssh に使うコマンドは SVN_SSH 環境変数で指定する

```
例１：set SVN_SSH=D:/Program Files/putty/plinkw.exe -l testuser -i g:/temp/hoge.key
例２：export SVN_SSH="ssh -i /home/user/.ssh/XXXXXX.ppk"
```

■ samba

```
svn checkout file:// ホスト /svnRepos
svn checkout file:///\ ホスト /svnRepos
svn checkout file:////\ ホスト /svnRepos
svn checkout file:\\\\ ホスト /svnRepos
```

などでアクセス可能・・・かも？

■ proxy

servers ファイルの global セクションを編集

```
http-proxy-host = server_name
http-proxy-port = port
```

サーバー

■とりあえず使う

リポジトリを初期化します

```
svnadmin create C:\home\repos
```

または

```
svnadmin create /home/svn/test --pre-1.5-compatible
```

などでバージョンを指定しても良い。
インポートしたいディレクトリ以下で

```
svn import file:/// リポジトリ -m "Initial import."
```

■ Subversion & ssh で外部からアクセス可能。

SSH サーバーを立てて、リポジトリを初期化しておくだけ。
クライアントからは

```
svn ls svn+ssh://[ ユーザー名 @]localhost/home/hoge/repo
```

■ Subversion & Webdav で外部からアクセス可能。

```
# svnadmin create --fs-type=fsfs /home/svnroot
# htpasswd -cm /home/svnroot/.htpasswd test
# chown -R apache:apache /home/svnroot
# chcon -R system_u:object_r:httpd_sys_content_t /home/svnroot


LoadModule dav_svn_module     modules/mod_dav_svn.so
LoadModule authz_svn_module   modules/mod_authz_svn.so

<Location /svn/repos>
   DAV svn
   SVNPath /home/svnroot

   # Limit write permission to list of valid users.
   <LimitExcept GET PROPFIND OPTIONS REPORT>
      # Require SSL connection for password protection.
      # SSLRequireSSL

      AuthType Basic
      AuthName "Authorization Realm"
      AuthUserFile /home/svnroot/.htpasswd
      Require valid-user
   </LimitExcept>
</Location>
```

ユーザーファイルの作成は

```
htpasswd -c /home/svnroot/.htpasswd user1
```

ユーザの追加は

```
htpasswd /home/svnroot/.htpasswd user2
```

```
<LimitExcept GET PROPFIND OPTIONS REPORT>
```

は、読込み以外は制限する意味。（読込みは認証を求めない）

■ svnserve を使う

http://d.hatena.ne.jp/kanonji/20090915/1253019999
基本的に Linux の場合は、svn+ssh で良い。
Linux で svnserve を使って ldap 認証を行うには svnserve で SASL 認証を参照

以下は、Windows を前提とする。

・設定ファイルの編集

C:\svnrepos\conf\svnserve.conf

```
[general]
anon-access = none
auth-access = write
password-db = passwd
```

C:\svnrepos\conf\passwd

```
foo = passwordAsPlainText
```

・サービスの登録

```
sc.exe create svnserve binpath= "C:\Program Files\Subversion\bin\svnserve.exe --service --root C:\svnrepos" displayname= "Subversion" depend= tcpip start= auto
```

コマンド　使い方

■チェックアウト

```
svn co リポジトリ チェックアウト先
svn co file:///svn/repo .
```

■アップデート

他の作業者がコミットした変更点を自分の作業コピーにマージする。

```
svn update
```

update 時に表示される記号の意味は以下の通り。

記号	意味
A	作業コピーに新しいファイルが追加された
D	作業コピーからファイルが削除された

U	作業コピーには何も手を加えていない状態でリポジトリから変更が行われた
G	作業コピーで変更を加えているファイルに対して、リポジトリから衝突が発生せずに変更が行われた
C	作業コピーで変更を加えているファイルに対して、リポジトリから変更が行われた際に衝突が発生した

■状態確認

作業コピーの状態を表示する

```
svn status
```

■作業内容を戻す

```
svn revert
```

作業コピーの内容を全て戻す場合は

```
svn revert -R .
```

■コミット

自分が変更した内容をリポジトリに登録する。

```
svn commit -m 'コミットメッセージ'
```

■ファイルやディレクトリを新規に追加する場合

普通に作業ディレクトリにファイルを追加する。その後

```
$ svn add hoge.txt
```

とする。ただしこの状態はあくまで予告なので本当に追加されるのは commit した時点である。

■ファイルやディレクトリを削除する場合

```
$ svn remove(rm / remove) hoge.txt
```

これも予告なので、commit 時に削除される。

■ファイルやディレクトリの名前を変更する場合

```
$ svn move hoge.txt hoge2.txt
```

即座に名前が変更されます。

■履歴表示

コミットの履歴を表示する。

```
svn log
```

全て表示せずに、表示する数を指定する場合

```
svn log -l 10
```

svn copy した時点のリビジョンまで表示する場合

```
svn log --stop-on-copy
```

■ブランチの作成

```
svn copy コピー元 コピー先
```

■マージ

例えばブランチの変更点を全て作業コピーにマージする場合

```
svn merge -r10:20 file://svn/repo/branch .
```

■マージの適用状況を確認

```
svn mergeinfo source target
```

例えば、ブランチの変更点が全て trunk にマージされているか確認

```
svn mergeinfo file://svn/repo/branch file://svn/repo/trunk
svn mergeinfo --show-revs merged file://svn/repo/branch file://svn/repo/trunk
```

逆にマージされていないリビジョンを表示する場合は

```
svn mergeinfo --show-revs eligible file://svn/repo/branch file://svn/repo/trunk
```

■ロック

```
svn lock test.txt
```

■ロックの強制

```
svn propset svn:needs-lock ON test.txt
svn propset -R svn:needs-lock ON *
```

■ロックの強制の解除

```
svn propdel svn:needs-lock test.txt
```

■アンロック

```
svn unlock test.txt
```

■ロックされているファイルの一覧

```
svnadmin lslocks ローカルリポジトリ
```

典型的なサイクル

■パターン１

開発が終了して保守期間での管理。

1. 修正毎にブランチを作成
2. 変更はブランチに対して行う
3. テスト前に、トランクの修正内容をブランチにマージ
4. テスト中の修正もブランチに行う
5. リリース時にブランチに内容をトランクにマージ
    1. ブランチにマージした時点のトランクとブランチの差をトランクへマージ
6. トランクのタグを作成

■以前のリビジョンに戻す

svn merge コマンドを使うと、反対向きの差分を指定して作業コピーの変更を「取り消す」ことができる。 以下はリビジョン 303 を破棄して 302 に戻す例である。

```
$ svn merge -r 303:302 http://svn.example.com/repos/calc/trunk
```

■接続時のエラーについて

http://www.nabble.com/svn:-Expected-FS-format-%272%27--found-format-%273%27-td18827391.html
http://plaza.rakuten.co.jp/locomassy/diary/?ctgy=2

```
svn: Expected FS format '2'; found format '3'
```

のようなエラーの場合､リポジトリを作成した SVN とクライアントのバージョンが合っていない可能性あり。

```
format 2： Subverion 1.4 以降でアクセス可能
format 3： Subverion 1.5 以降でアクセス可能
```

```
svnadmin create /home/svn/test --pre-1.5-compatible
```

でリポジトリ作成時のバージョンを指定する良い。

その他

■―stop-on-copy オプション

ブランチが作成されたリビジョン（ブランチのベースリビジョンという）を探すには svn log コマンドで ―stop-on-copy オプションを利用する

■あるリビジョン以降で変更されたファイルの一覧

```
svn diff -r 100:head --summarize
```

このリストのファイルを他のディレクトリにコピーしたい場合は
を使う。

```
svn diff -r 100:head --summarize > list.txt
copySVN list.txt todir
```

■特定のディレクトリ以下の更新を無視する

```
$ svn update --set-depth exclude path
```

戻すには

```
$ svn update --set-depth infinity
もしくはパスが正確にわかっているなら
$ svn update path
```

exclude 以外に以下の depth オプションがある。

--	--
exclude	対象ディレクトリを更新しない
empty	対象ディレクトリの中身を更新しない
files	対象ディレクトリの中にあるディレクトリを更新しない (ファイルのみ更新する)
immediates	対象ディレクトリ内のファイルとディレクトリは更新するが、対象ディレクトリ内のディレクトリの中身は更新しない
infinity	再帰的に全て更新する

■ローカルにファイルを残したまま、管理から外す

```
svn delete build --keep-local
```

■ SVN でログメッセージを変更する

```
$ svn propset --revprop -r リビジョン svn:log " ログメッセージ "
```

ログの内容をファイルで指定する場合は

```
$ svn propset --revprop -r リビジョン svn:log -F log.txt
```

出来ない場合は、設定が必要。
SVN でログメッセージを変更する

■ patch を作成、 patch を当てる

```
svn diff -rxxx:yyyy > patch
patch -p0 -l < patch
```

■ merge 属性を消す

```
svn propdel -R svn:mergeinfo .
```

■ svn status や svn merge --dry-run 実行時に表示される記号の意味

http://code1.jldg.org/trac/bridge/wiki/svn_trouble

記号	内容
U	ファイルがリポジトリの最新版に更新された
A	ファイルが新規追加された
D	ファイルが削除された
R	ファイルが置き換えられた（同じ名前だが履歴上は別物）
G	ローカルの修正とリポジトリの更新がマージされた
C	ローカルの修正とリポジトリの更新が競合している

■ merge の後の commit で、tree-conflict がでて commit できない

tree conflict は、ファイルの存在に関する衝突です。 ローカルで消したファイルに変更点がある時などに発生します。
tree conflict が発生したファイルには、自動で差分を当ててくれないので、変更を自分で当ててください。 最後に、ファイルの変更点などを確認して、問題がなければ

```
$ svn resolve --accept working filename
```

で conflict の解消宣言をすれば、commit できるようになります。

■ svn diff に -x のオプションを複数つける

```
svn diff -x "-b --ignore-eol-style" Hello.cpp
```

■ svn diff で全てのスペースを無視する

```
svn diff -x "-w --ignore-eol-style" Hello.cpp
```

■ svn diff で改行コードを無視する

```
diff --strip-trailing-cr file1 file2
```

■ブランチの派生元を調べる

```
svn log -v
```

でログの詳細が見れる。これで、ブランチの派生元も分かる。

```
svn log -v --stop-on-copy
```

■ファイル単位でマージする

ファイルを指定すればファイル単位でマージできる

```
svn merge -r20:25 test.txt
```

もし、ブランチで追加されたファイルをファイル単位でマージする場合は、svn cp でブランチのファイルをコピーする

```
svn cp svn://hoge/added.txt ./added.txt
```

■ svn で他のパスにリンクを貼る

```
svn propset svn:externals 'hoge file://hogehoge/foo/bar'
```