Taddong

Things to take in account about SVN

Model	Copy-Modify-Merge
Syntax	Repositories addresses are urls

References

[1]	Version Control with Subversion	Ben Collins-Sussman	http://svnbook.red-bean.com/
		Brian W. Fitzpatrick	
		C. Machael Pilato	

This cheatsheet is basically a compilation of reference [1]

Working Copies

Definition	Is a local copy of the repository, in the form of an ordinary directory tree on the local file system.			
	You can work with these files exactly as if there were just local.			
	Subversion will never incorporate other people's changes, nor make your own changes available to others, until you explicitly tell it to do so.			
	Subversion provide commands to "publish" your changes or to merge others' changes into your working copy by reading the repository.			
Extra files	.svn directory	Working Copy administrative directory	Created and maintained by Subversion	
			Each directory of the cworking copy contains one	
			Help Subversion keep track of changes	
			A typical working copy usually corresponds to a particular subtree of the respository, because a repository usually contains many projects	
Get a working copy	You must check out some sobtree of the repository	**svn checkout http://svn.example.com/repos/calc**		
			A calc/Makefile A calc/integer.c A calc/button.c	'A' means subversion adding an item to your working copy
			Checked out revision 56.	Summary of the command and the revision checked out.
Publish your changes	You must commit your changes	**svn commit button.c -m "Fixed a typo in button.c."**		-m send a description of your changes.
			Sending button.c Transmitting file data . Committed revision 57.	Subversion commiting your changes to the reporitory
Bring your project up to date	You must update your working copy	**svn update**		
			U button.c	File(s) being updated
			Updated to revision 57.	Summary of the command and the updated revision.

Revisions

Definition	Each time the repository accepts a commit, this creates a new state of the filesystem tree, called a revision.
	Each revision is assigned a unique natural number, one greater than the number of the previous revision.
	The initial revision of a freshly created repository is numbered 0 and consists of nothing but an empty root directory.
	Subversion's revision numbers apply to entire trees, not individual files.

How Working Copies Track the Repository

For each file in a directory	.svn directory keeps track of	What revision your working file is based on		This is called the file's working revision
		A timestamp recording when the local copy was last updated by the repository		
States of a working file	Unchanged and current	The file is unchanged in the working directory, and no changes to that file have been committed to the repository since its working revision.		
		An 'svn commit' of the file will do nothing, and an 'svn update' of the file will do nothing.		
	Locally changed, and current	The file has been changed in the working directory, and no changes to that file have been committed to the repository since you last updated.		
		Thus an 'svn commit' of the file will succeed in publishing your changes, and an 'svn update' of the file will do nothing.		
	Unchanged, and out of date	The file has not been changed in the working directory, but it has been changed in the repository. The file might be updated to make it current to last rev.		
		An 'svn commit' of the file will do nothing, and an 'svn update' of the file will fold the latest changes into your working copy.		
	Locally changed, and out of date	The file has been changed both in the working directory and in the repository.		
		An 'svn commit' of the file will fail with an "out-of-date" error. The file should be updated first.		
		an 'svn update' command will attempt to merge the public changes with the local changes.		
			If Subversion can't complete the merge in a plausible way automatically, it leaves it to the user to resolve the conflict.	
Show state	Of a working item	**svn status**	This command will show you the state of any item in your working copy.	

Mixed Revisions Working Copies

Principle	The ability to have a working copy containing files and directories with a mix of different working revision numbers.			
Updates and Commits	A "push" action does not cause a "pull", nor vice versa.			
	If you have new changes still in progress, 'svn update' should gracefully merge repository changes into your own, rather than forcing you to publish them.			
	Example:	You have a working copy entirely at revision 10		
		You edit foo.html and perform 'svn commit'		This creates a version 15 (for example) in the repository
			Working Copy isn't at revision 15!! (any number of changes might have happened in the repository between revisions 10 and 15.)	
			You haven't tun 'svn update' and 'svn commit' do NOT pull the changes between 10 and 15 revisions.	
			The only safe thing the Subversion client can do is mark the one file—foo.html—as being at revision 15.	
			The rest of the working copy remains at revision 10.	
			Only by running svn update can the latest changes be downloaded and the whole working copy be marked as revision 15.	
History of Changes	History	**svn log**	This command will display the history of changes to a file or directory.	
	Examine Mixture	**svn status --verbose**	Examine the detail of mixture of different versions	

Taddong

Initial checkout

checkout	svn chekout <url> [directory]	
		The copy contains the HEAD (latest version) of the subversion repository specified in the url.
		You can checkout the main trunk or a subdirectory of it.
		You can specify a directory where subversion will put the trunk

Getting Data into your repository

import	svn import	
		A quick way to copy an unversioned tree of files into a repository, creating intermediate directories as necessary.
		This command doesn't require a working copy, and your files are immediately committed to the repository.

Basic Working Cycle

1.- Update your working copy	Receive other's changes			
	svn update			Bring your working copy into sync with the latest revision in the repository
	svn help update			Help on the update command
2.- Make changes	To edit a file you don't need to type any command			
	You can make file changes or tree changes			
	svn add <foo>			Schedule file, directory, or symbolic link foo to be added to the repository.
	svn delete <foo>			Schedule file, directory, or symbolic link foo to be deleted from the repository.
	svn copy <foo> <bar>			Create a new item bar as a duplicate of foo and automatically schedule bar for addition.
	svn move <foo> <bar>			This command is exactly the same as running svn copy foo bar; svn delete foo.
	svn mkdir <blort>			This command is exactly the same as running mkdir blort; svn add blort.
3.- Examine your changes	svn status [-v] [-u] [<path>]			Get an overview of your changes. The results of this command are relative to your current directory
		First results column: status of a file or directory		
		A		The file, directory, or symbolic link item has been scheduled for addition into the repository.
		C		The file item is in a state of conflict. Changes received from the server during an update overlap with local changes that you have in your working copy.
		D		The file, directory, or symbolic link item has been scheduled for deletion from the repository.
		M		The contents of the file item have been modified.
		The verbose (-v) option will show you the status of every item of your working copy, even if it hasn't changed.		
		Column #2		Working revision of the item
		Column #3		The revision in which the item was last changed
		Column #4		Who changed it
		The show-updates option (-u) contacts the repository and adds information about thing that are out of date		
	svn diff	Examine the details of your local modifications, printing them in unified diff format.		
		You can generate a patch by redirecting the output to a patch file (usable by patch command)		
4.- [undo some working changes]	svn revert <item>			Reverts item to its premodified state, including any operation (addition, copy, etc.)
5.- Resolve conflicts		When conflicts appear, 'svn update' command presents some options		
			(p) postpone	
			(df) diff-full	
			(e) edit	
			(r) resolved	
			(mf) mine-full	
			(tf) theirs-full	
			(l) launch	
			(h) help	
		For every postponed conflicted file, Subversion places three extra unversioned files in your working copy:		
			filename.mine	This is your file as it existed in your working copy before you updated your working copy.
				If Subversion considers the file to be unmergeable, the .mine file isn't created)
			filename.rOLDREV	This was the BASE file revision before you updated your working copy (the one that you checked out before you made your edits)
			filename.rNEWREV	The file that your svn client just received from the server when you updated your working copy (= repository's HEAD revision)
	svn resolve --accept [arg] <item>			Resolves the conflicts found in <item> as you specify and removes the three versions of the item
		Where 'arg'		
			base	To choose the version of the file that you last checked out before making your edits.
			mine-full	To choose the version that contains only your edits
			theirs-full	To choose the version that your most recent update pulled from the server (and thus discarding your edits entirely)
			working	Merge the conflicted text “by hand”, by editing your working copy
6.- Commit your changes	svn commit [-m "Message" | -f File]			The svn commit command sends all of your changes to the repository.
		"Message"		A log message for yoour changes
		File		A file where the log message is taken from