

VCS — Subversion Support

Command	Keystroke	Function	Note
Emacs Support for Version Control Subversion Subversion home Subversion book book in html book in pdf	- Emacs has built-in support for Version Control Systems through its VC interface that supports several popular VCS backends and works well for most simple operations. - VC supports Subversion, using the same VC commands that are specialized for Subversion. - Emacs VC detects that the file is under Subversion revision control and activates its Subversion backend. - Emacs VC Subversion backend works with Subversion prior and after version 1.7: it detects which <code>.svn</code> directory must be used. - There are also several external packages that provide extended, specialized support for Subversion:  The psvn external package.  PEL activates it when the pel-use-psvn user-option is turned on (set to t).  dsvn external package.  PEL activates it when the pel-use-psvn user-option is turned on (set to t).		
Open this PDF file. See also: 🔗 Help/Info	<f11> v <f1> <f12> <f1>	(pel-help-pdf &optional OPEN-WEB-PAGE)	Prompt to open one of the VCS PDF files like the 🔗 VCS-Mercurial local PDF. If the prefix argument (like C-u or M--) is used, then it opens the remote GitHub hosted raw PDF instead. If the pel-flip-help-pdf-arg user-option is set it's the other way around.
🔗 Customize PEL VCS control	<f11> v <f2> <f12> <f2>	(pel-customize-pel &optional OTHER-WINDOW)	Customize PEL Version Control System support. If OTHER-WINDOW is non-nil (use C-u) , display in other window.
🔗 Customize Emacs VCS control	<f11> v <f3>	(pel-customize-library &optional OTHER-WINDOW)	Customize Emacs Version Control System support: vc, vc-hg, vc-git, magit, monkey.
	<f12> <f3>		Customize Emacs Version Control System support: vc, vc-hg, vc-git.
Emacs Built-in VCS package: VC	Emacs built-in VCS support is provided by the VC library. - The VC library supports several VCS tools identified by the <code>vc-handled-backends</code>, which identifies: RCS and SCCS. CVS and Subversion, Bazaar, Git, Mercurial and Monotone. - When visiting a directory or a file and using a VC command to manage the repository, VC automatically detects the (D)VCS type for the repository, set the variable <code>vc-dir-backend</code> and adjusts its backend accordingly. The following commands can be use to force the use of another VC backend if the files are available (like the <code>.git</code> or <code>.hg</code> directories for Git and Mercurial respectively).		
Detect available VCS backend and switch to one selected	<f11> v s	(pel-vcs-switch-backend)	Switch VCS back-end for the current file. - If no VCS back-end or only one VCS back-end is available for the file, the command issues an error, otherwise it prompts for the VCS back-end to use and switch the VCS back-end for this file. The switch is temporary: it is restricted to the current Emacs session. - Use this command when the file is managed by more than one VCS back-end. - This uses the function 'vc-switch-backend' to perform the switch.
Switch VCS backend	C-x v b	(vc-switch-backend FILE BACKEND)	Make BACKEND the current version control system for FILE. - FILE must already be registered in BACKEND. The change is not permanent, only for the current session. The function only changes VC's perspective on FILE, it does not register or unregister it. - By default, this command cycles through the registered backends. - To get a prompt, use a prefix argument.
Open the VCS Status Window	With VC, you manage the repository through a buffer using the vc-dir command opens a <code>*vc-dir*</code> buffer that uses the <code>vc-dir-mode</code> major mode. - The buffer shows the status of files unregistered, modified and not yet committed. - The buffer supports a set of commands, mostly single character commands to quickly perform operations on the repository. 👉 These keys are shown in class with light blue background beside globally bound commands that can be issued from any buffer.		
Type ? for help.	For a list of the the key bindings available in the <code>*vc-dir*</code> buffer use the describe-mode command: (type ? in the buffer's window).		
Open a VC Status buffer	- C-x v d <f11> v v	(vc-dir DIR &optional BACKEND)	Open a <code>*vc dir*</code> buffer that shows the Subversion status of files in the current directory tree. 👉 Type C-u prefix to request a different VC backend: the command prompts for the backend allowing you to select a different backend than the automatically selected one.
List status of files in <code>*vc dir*</code> buffer svn status svn status man page		Show the VC status for "interesting" files in and below DIR. - This allows you to mark files and perform VC operations on them. - The list omits files which are up to date, with no changes in your copy or the repository, if there is nothing in particular to say about them. - Preparing the list of file status takes time; when the buffer first appears, it has only the first few lines of summary information. The file lines appear later. ⚠ This command can take a long time for a large directory tree that's a (or part of) a Subversion repository. - However it does not block Emacs; you can continue to work in other buffers. 👉 With Subversion it's often better to open the repo down into the directory where the changes occurred, specially if the directory is several levels down a large Subversion repository directory tree.	
Ignoring Files	The following command can be used to add files that must be ignored by Subversion.		
Ignore a file	C-x v G	(vc-ignore FILE &optional DIRECTORY REMOVE)	Ignore FILE under the VCS of DIRECTORY. Interactively prompts for the file name. - Normally, FILE is a wildcard specification that matches the files to be ignored. When REMOVE is non-nil, remove FILE from the list of ignored files. - DIRECTORY defaults to 'default-directory' and is used to determine the responsible VC backend. - When called interactively, prompt for a FILE to ignore, unless a prefix argument is given, in which case prompt for a file FILE to remove from the list of ignored files.
	G	(vc-dir-ignore)	Inside <code>*vc-dir*</code> buffer: Ignore the <i>current</i> file. This does noting in Subversion.
Hiding vc-dir lines	You can hide files that are in specific state (like unregistered, or ignored) by using the following command.		
Hide specific states	H	(pel-vc-dir-hide STATES)	Hide lines corresponding to specified STATES. Prompt for state with tab completion. Press g to refresh list and display all states again.
Log svn commands	Log the VC backend command into the <code>*pel-vc-log*</code> buffer. The events show the event as lisp lists.		
Toggle logging VC backend commands	<f11> v l	(pel-vcs-toggle-vc-log)	Start/stop logging VC commands in the <code>*pel-vc-log*</code> buffer. When starting, does not create the buffer. It is created on the first VC event.
VC commands for Subversion Subversion book: html , pdf	Emacs VC mode supports Subversion when the <code>svn</code> command is available. No special setup is required, VC will automatically detect that the file is inside a Subversion repository: it finds the appropriate <code>.svn</code> directory. When editing a file that is inside a Subversion repository, you can use the following VC commands.		
Add/commit a file	If a file is file is not under Subversion control, you must first add it. Once it is under Subversion control then you can commit it. - Inside the <code>*vc-dir*</code> buffer, to add file first select them (with the m key) and then the v key to add them. - The command will be rejected if files currently managed are selected. Unselect them with the u or U key first then try again. - Once a file is managed by Subversion, you can select its name and commit it by using the v key again.		
Add the file to the repo	C-x v i	(vc-register &optional VC-FILESET COMMENT)	Register into a version control system: add the file to Subversion control.
Add/Commit current file	C-x v v	(vc-next-action VERBOSE)	Executes, for the current file (or all marked files in the <code>*vc-dir*</code> buffer): - add or commit the files - Refuses to perform the action when state of <code>*vc-dir*</code> selected items differ.
Add/Commit selected file(s)	v		The command is also available by typing v inside the *vc-dir* buffer. It affects all currently selected files.

Command	Keystroke	Function	Note
• Annotate a file	List changes in files, showing the revision id responsible for each line. - This command is useful for discovering when a change was made and by whom. 👉 Once you are inside the *Annotate file (rev #)* buffer you can easily navigate through relevant revisions, show other diffs and file content at specific revisions. This is extremely useful with Subversion because it uses revision numbers that are global and change every time a new commit is done. See Subversion Fundamental Concepts - Version Control the Subversion Way for more info.		
Annotate version of each line of file	C-x v g	(vc-annotate FILE REV &optional DISPLAY-MODE BUF MOVE-POINT-TO VC-BK)	Annotate the lines of the current file: open a *Annotate file (rev #)* buffer that shows the annotation of each line of the current file, each changes has its own background colour. - More commands are available in the *Annotate* buffer. 👉 This is very useful in Subversion: see below
• Annotation buffer visibility	Inside the *Annotate file (rev #)* , the following keys are bound to visibility control commands.		
Toggle visibility of annotation info.	v	(vc-annotate-toggle-annotation-visibility)	Toggle visibility of the annotation information shown at the left of each line in the annotation buffer: the Subversion revision number and the author's name. Line colours are not changed.
• Annotate other revisions	The vc-annotate command on a file opens a *Annotate file (rev #)* that shows all lines, coloured by revision and identifying the author for each. The buffer supports an extra set of very useful functions listed below.		
Annotate the revision before revision of current annotated line.	a	(vc-annotate-revision-previous-to-line)	Visit the annotation of the revision before the revision at line.
Visit Annotation of revision at current line	j	(vc-annotate-revision-at-line)	Visit the annotation of the revision identified in the current line.
Annotate the next revision	n	(vc-annotate-next-revision PREFIX)	Visit the annotation of the revision after this one. With a numeric prefix argument, annotate the revision that many revisions after.
Annotate the previous revision	p	(vc-annotate-prev-revision PREFIX)	Visit the annotation of the revision previous to this one. With a numeric prefix argument, annotate the revision that many revisions previous.
Annotate the working directory revision.	w	(vc-annotate-working-revision)	Visit the annotation of the working revision of this file. 👉 Useful to get back to where you started from, since there can be several subversion versions in between changes of a given file (because all files inside the entire Subversion repo have the same version number and that number changes each time someone commits something in the Subversion repo).
• Visit diff of revs	Inside the *Annotate file (rev #)* , the following keys are bound to operations that open buffers with the diff between versions identified by the current line in the *Annotate file (rev #)* buffer.		
Opens the complete change-set corresponding to the current line.	D	(vc-annotate-show-changeset-diff-revision-at-line)	Visit the diff of the revision at line from its previous revision for all files in the changeset. Opens another buffer to display this information.
Visit the diff of the rev/prev-rev at line	= d	(vc-annotate-show-diff-revision-at-line)	Visit the diff of the revision at line from its previous revision.
• Visit file content of rev	Inside the *Annotate file (rev #)* , the following keys are bound to operations that open buffers with the content of file at the version identified by the current line in the *Annotate file (rev #)* buffer.		
Goto corresponding line in versioned file buffer	RET	(vc-annotate-goto-line)	Go to the line corresponding to the current VC Annotate line.
Visit the file at revision of current line	f	(vc-annotate-find-revision-at-line)	Visit the revision identified in the current line.
• Visit logs	Inside the *Annotate file (rev #)* , the following keys are bound to operations to extra logs from a revision identified at a line in the buffer.		
Visit log of revision at line	l	(vc-annotate-show-log-revision-at-line)	Visit the log of the revision at line. - If the VC backend supports it, only show the log entry for the revision. - If a *vc-change-log* buffer exists and already shows a log for the file in question, search for the log entry required and move point.
• Scrolling annotate buffer	Inside the *Annotate file (rev #)* , the following keys are bound to buffer scroll operations. See § Scrolling for more scrolling commands.		
Scroll up by near full screen	SPC C-v	(scroll-up-command &optional ARG)	Scroll text of selected window upward ARG lines; or near full screen if no ARG. - If 'scroll-error-top-bottom' is non-nil and 'scroll-up' cannot scroll window further, move cursor to the bottom line. When point is already on that position, then signal an error. - A near full screen is 'next-screen-context-lines' less than a full screen. - Negative ARG means scroll downward. - If ARG is the atom '-1', scroll downward by nearly full screen.
Scroll down by near full screen	DEL S-SPC M-v	(scroll-down-command &optional ARG)	Scroll text of selected window down ARG lines; or near full screen if no ARG. - If 'scroll-error-top-bottom' is non-nil and 'scroll-down' cannot scroll window further, move cursor to the top line. When point is already on that position, then signal an error. - A near full screen is 'next-screen-context-lines' less than a full screen. - Negative ARG means scroll upward. - If ARG is the atom '-1', scroll upward by nearly full screen.
• Move in annotate buf	Inside the *Annotate file (rev #)* , the following keys are bound to navigation commands. See § Navigation for more navigation commands.		
Move point to top of buffer	< M-<	(beginning-of-buffer &optional ARG)	Move point to the beginning of the buffer. - With numeric arg N, put point N/10 of the way from the beginning. - If the buffer is narrowed, this command uses the beginning of the accessible part of the buffer. - Push mark at previous position, unless either a C-u prefix is supplied, or Transient Mark mode is enabled and the mark is active.
Move point to bottom of buffer	> M->	(end-of-buffer &optional ARG)	Move point to the end of the buffer. - With numeric arg N, put point N/10 of the way from the end. - If the buffer is narrowed, this command uses the end of the accessible part of the buffer. - Push mark at previous position, unless either a C-u prefix is supplied, or Transient Mark mode is enabled and the mark is active.
• Other commands	Other commands available in the *Annotate file (rev #)* buffer:		
Revert buffer	g	(revert-buffer &optional IGNORE-AUTO NOCONFIRM PRESERVE-MODES)	Replace current buffer text with the text of the visited file on disk. - This undoes all changes since the file was visited or saved. - With a prefix argument, offer to revert from latest auto-save file, if that is more recent than the visited file.
Quit window	q	(quit-window &optional KILL WINDOW)	Quit WINDOW and bury its buffer. - WINDOW must be a live window and defaults to the selected one. - With prefix argument KILL non-nil, kill the buffer instead of burying it.
View Mode commands	? h	(describe-mode &optional BUFFER)	Display documentation of current major mode and minor modes. A brief summary of the minor modes comes first, followed by the major mode description. This is followed by detailed descriptions of the minor modes, each on a separate page.

Command	Keystroke	Function	Note
Display Revision Log See also: ↗ Diff & Merge	The VC-mode for Subversion provide the following commands to open buffer that shows the Subversion log of a specific file. - Subversion logs are show inside *vc-change-log* buffers that operate in the vc-svn-log-view-mode. The log can be opened when visiting a file or the annotation of a file (see *Annotate file (rev #)* commands in the previous section). The commands to open the repo or file log are shown below followed by commands available in the *vc-change-log* buffers 👁️ When looking at log you may want to see the list of modified files inside each log entry. For that change the following user-option: vc-svn-global-switches : you could add the <code>--verbose</code> switch to the list to see the names of the files when printing the log. Unfortunately several svn commands do not accept that switch and the command will fail. 👉 PEL provides the pel-vcs-svn-verbose-log user option: when this is turned on PEL activates the <code>--verbose</code> switch for the commands that print subversion log history. That does not affect the other commands. 👉 The vc-print-root-log command prints a log history for each of the files modified included in the committed file set, not only the current file. - It's useful to dig into the changes for all files part of this submission: use log-view-diff or log-view-diff-changeset to see the changes. And then from the *vc-diff* buffer use diff-goto-source to visit the current version of that file. - From the *vc-diff* buffer you can see its own log and also see the differences. See ↗ Diff & Merge for more info.		
Print repo log: includes each revision in the repo history 🍀: pel-vcs-svn-verbose-log	C-x v L	(vc-print-root-log &optional LIMIT)	List the revision history for the current VC controlled tree in a window. - If LIMIT is non-nil, it should be a number specifying the maximum number of revisions to show; the default is 'vc-log-show-limit'. - When called interactively with a prefix argument, prompt for LIMIT. - When the prefix argument is a number, use it as LIMIT. - A special case is when the prefix argument is 1: in this case the command asks for the ID of a revision, and shows that revision with its diffs (if the underlying VCS supports that).
Open the change log for the file Includes only changes for the current file 🍀: pel-vcs-svn-verbose-log	C-x v 1	(vc-print-log &optional WORKING-REVISION LIMIT)	List the change log of the current fileset in a window. - If WORKING-REVISION is non-nil, leave point at that revision. - If LIMIT is non-nil, it should be a number specifying the maximum number of revisions to show; the default is 'vc-log-show-limit'. - When called interactively with a prefix argument, prompt for WORKING-REVISION and LIMIT.
	1		In the *vc-dir* buffer
Annotation buffer visibility	Inside a *vc-change-log* buffer the following command can be used to highlight the log message for the version at point. - Several of version comments section can be marked/highlighted with the command. - Aside from highlighting the version that does not seem to be used anywhere.		
Toggle marked state of log entry at point	m	(log-view-toggle-mark-entry)	Toggle the marked state for the log entry at point. - Individual log entries can be marked and unmarked. - The marked entries are denoted by changing their background color. 'log-view-get-marked' returns the list of tags for the marked log entries.
Annotate other revisions	The log-view-annotate-version command on a file opens a *Annotate file (rev #)* that shows all lines, coloured by revision and identifying the author for each. The buffer supports an extra set of very useful functions listed below. See the commands available in that type of buffer in the previous page. 👉 This command is very useful because from the annotate buffer you can annotate an other older version or visit the content the file had for that version: using this command allows you to navigate through several older versions of the file.		
Annotate revision identified at current line	a	(log-view-annotate-version POS)	Annotate the version at POS. If called interactively, annotate the version at point.
Visit file content of rev	Inside the *vc-change-log*, the following keys are bound to operations that open buffers with the content of file at the version identified by the current line of *vc-change-log* buffer.		
Visit the content of the file at specified revision	f	(log-view-find-revision POS)	Visit the version at POS. If called interactively, visit the version at point.
Visit diff of revs	Inside the *vc-change-log*, the following keys are bound to operations that open buffers with the diff between versions.		
View diff between 2 revisions of the current file	= d	(log-view-diff BEG END)	Get the diff between two revisions. - If the region is inactive or the mark is on the revision at point, get the diff between the revision at point and its previous revision. Otherwise, get the diff between the revisions where the region starts and ends. - Unlike 'log-view-diff-changeset', this function only shows the part of the changeset which affected the currently considered file(s).
View diff between 2 revisions of all files in the current change-set.	D	(log-view-diff-changeset BEG END)	Get the diff between two revisions. - If the region is inactive or the mark is on the revision at point, get the diff between the revision at point and its previous revision. Otherwise, get the diff between the revisions where the region starts and ends. - Unlike 'log-view-diff' this function shows the whole changeset, including changes affecting other files than the currently considered file(s). 👉 For a single changed file that command might highlight line termination issues more than the log-view-diff command.
Log message operations	Inside the *vc-change-log*, the following keys are bound to operations that manipulate the log message view (and possibly the message).		
Expand the current Log View	RET	(log-view-toggle-entry-display)	If possible, expand the current Log View entry. This calls 'log-view-expanded-log-entry-function' to do the work.
Change version commit message (requires admin rights)	e	(log-view-modify-change-comment)	Edit the change comment displayed at point. To use this you must have admin rights to the depot since modification of the commit messages are not enabled by default but can be. See this Subversion FAQ topic.
Scroll vc-change-log buffer	Inside the *vc-change-log*, the following keys are bound to buffer scroll operations. See ↗ Scrolling for more scrolling commands.		
Scroll up by near full screen	SPC C-v	(scroll-up-command &optional ARG)	Scroll text of selected window upward ARG lines; or near full screen if no ARG. - If 'scroll-error-top-bottom' is non-nil and 'scroll-up' cannot scroll window further, move cursor to the bottom line. When point is already on that position, then signal an error. - A near full screen is 'next-screen-context-lines' less than a full screen. - Negative ARG means scroll downward. - If ARG is the atom '-1', scroll downward by nearly full screen.
Scroll down by near full screen	DEL S-SPC M-v	(scroll-down-command &optional ARG)	Scroll text of selected window down ARG lines; or near full screen if no ARG. - If 'scroll-error-top-bottom' is non-nil and 'scroll-down' cannot scroll window further, move cursor to the top line. When point is already on that position, then signal an error. - A near full screen is 'next-screen-context-lines' less than a full screen. - Negative ARG means scroll upward. - If ARG is the atom '-1', scroll upward by nearly full screen.
Move in vc-change-log buf	Inside the *vc-change-log*, the following keys are bound to navigation commands. See ↗ Navigation for more navigation commands.		
Move point to next diff	TAB n	(log-view-msg-next &optional COUNT)	Go to the next COUNT'th log message. Interactively, COUNT is the prefix numeric argument, and defaults to 1.
Move point to previous diff	<backtab> p	(log-view-msg-prev &optional COUNT)	Go to the previous COUNT'th log message. Interactively, COUNT is the prefix numeric argument, and defaults to 1.
Move point to next file diff	N M-n	(log-view-file-next &optional COUNT)	Go to the next COUNT'th file. Interactively, COUNT is the prefix numeric argument, and defaults to 1.
Move point to previous file diff	P M-p	(log-view-file-prev &optional COUNT)	Go to the previous COUNT'th file. Interactively, COUNT is the prefix numeric argument, and defaults to 1.

Command	Keystroke	Function	Note
Move point to top of buffer	< - M-<	(beginning-of-buffer &optional ARG)	Move point to the beginning of the buffer. - With numeric arg N, put point N/10 of the way from the beginning. - If the buffer is narrowed, this command uses the beginning of the accessible part of the buffer. - Push mark at previous position, unless either a C-u prefix is supplied, or Transient Mark mode is enabled and the mark is active.
Move point to bottom of buffer	> - M->	(end-of-buffer &optional ARG)	Move point to the end of the buffer. - With numeric arg N, put point N/10 of the way from the end. - If the buffer is narrowed, this command uses the end of the accessible part of the buffer. - Push mark at previous position, unless either a C-u prefix is supplied, or Transient Mark mode is enabled and the mark is active.
• Other commands	Other commands available in the *vc-change-log* buffer:		
Revert buffer	g	(revert-buffer &optional IGNORE-AUTO NOCONFIRM PRESERVE-MODES)	Replace current buffer text with the text of the visited file on disk. - This undoes all changes since the file was visited or saved. - With a prefix argument, offer to revert from latest auto-save file, if that is more recent than the visited file.
Quit window	q	(quit-window &optional KILL WINDOW)	Quit WINDOW and bury its buffer. - WINDOW must be a live window and defaults to the selected one. - With prefix argument KILL non-nil, kill the buffer instead of burying it.
View Mode commands	? - h	(describe-mode &optional BUFFER)	Display documentation of current major mode and minor modes. A brief summary of the minor modes comes first, followed by the major mode description. This is followed by detailed descriptions of the minor modes, each on a separate page.
• Comparing Versions See also: Diff & Merge	Use the following commands to compare the content of revisions of files. Some of the commands are also available from the *vc-dir* buffer. These are shown with key-bindings in the light-blue cells.		
Show the content of a specific revision of a file	C-x v ~	(vc-revision-other-window REV)	Visit revision REV of the current file in another window. - If the current file is named 'F', the revision is named 'F.~REV~'. - If 'F.~REV~' already exists, use it instead of checking it out again.
Diff versions of all files in directory	C-x v D	(vc-root-diff HISTORIC &optional NOT-URGENT)	Display diffs between VC-controlled whole tree revisions. - Normally, this compares the tree corresponding to the current fileset with the working revision. - With a prefix argument HISTORIC, prompt for two revision designators specifying which revisions to compare. - The optional argument NOT-URGENT non-nil means it is ok to say no to saving the buffer.
	D		
Diff versions of current file	C-x v =	(vc-diff &optional HISTORIC NOT-URGENT)	Show differences between current file version in repo and local file's content. - Display diffs between file revisions. - Normally this compares the currently selected fileset with their working revisions. - With a prefix argument HISTORIC (use C-u), it reads two revision designators specifying which revisions to compare. - In Elisp code: the optional argument NOT-URGENT non-nil means it is ok to say no to saving the buffer.
	=		
Update Repo	C-x v +	(vc-update &optional ARG)	Update the current fileset or branch. - You must be visiting a version controlled file, or in a 'vc-dir' buffer. - For each unchanged and unlocked file, this simply replaces the work file with the latest revision on its branch. If the file contains changes, any changes in the tip revision are merged into the working file.
• Reverting & Deleting			
Revert Files to copy in Subversion • svn revert	C-x v u	(vc-revert)	Revert working copies of the selected fileset to their repository contents. This asks for confirmation if the buffer contents are not identical to the working revision (except for keyword expansion).
Delete file from repo	C-x v x	(vc-delete-file FILE)	Delete file and mark it as such in the version control system. If called interactively, read FILE, defaulting to the current buffer's file name if it's under version control.
• Merging			
Merge	C-x v m	(vc-merge)	Perform a version control merge operation. - You must be visiting a version controlled file, or in a 'vc-dir' buffer. - This asks for two revisions to merge from in the minibuffer. - If the first revision is a branch number, then merge all changes from that branch. - If the first revision is empty, merge the most recent changes from the current branch.
• Tag Management			
Retrieve by tag	C-x v r	(vc-retrieve-tag DIR NAME)	For each file in or below DIR, retrieve their tagged version NAME. - NAME can name a branch, in which case this command will switch to the named branch in the directory DIR. - Interactively, prompt for DIR only for VCS that works at file level; otherwise use the repository root of the current buffer. - If NAME is empty, it refers to the latest revisions of the current branch. - If locking is used for the files in DIR, then there must not be any locked files at or below DIR (but if NAME is empty, locked files are allowed and simply skipped). - This function runs the hook 'vc-retrieve-tag-hook' when finished.
Create a tag	C-x v s	(vc-create-tag DIR NAME BRANCHP)	Descending recursively from DIR, make a tag called NAME. - For each registered file, the working revision becomes part of the named configuration. - If the prefix argument BRANCHP is given, the tag is made as a new branch and the files are checked out in that new branch.
• Other Commands			
Switch VC Backend	C-x v b	(vc-switch-backend FILE BACKEND)	Make BACKEND the current version control system for FILE. - FILE must already be registered in BACKEND. The change is not permanent, only for the current session. This function only changes VC's perspective on FILE, it does not register or unregister it. - By default, this command cycles through the registered backends. - To get a prompt, use a prefix argument.

Subversion Support - References

Topic	Notes
Subversion	Subversion Version 1.7 is a major change: - Before svn 1.7 there was one .svn subdirectory per directory. Starting with svn 1.7 there is only one .svn subdirectory. - Also in subversion 1.7 the output of modified externals has changed to include the full path rather than the relative path.
Apache Subversion @ Wikipedia	Provides a good overview. Start here.

Topic	Notes
Apache Subversion home page	
Subversion Book	Link to the page where various versions of the Subversion books available. Here are additional links: - Subversion Book, version 1.7, multiple-html pages TOC - Subversion Book, version 1.4, multiple-html pages TOC
Emacs Subversion Support	Emacs supports Subversion through Emacs VC-mode (Version Control mode)
Emacs VC - Version Control support	The Emacs page describing Emacs VC mode.
Subversion @ Emacs Wiki	Describes Emacs Subversion support through Emacs VC. - Subversion backend is activated by the presence of the .svn directories. - Diff between two files can be done by Subversion using the defined diff-cmd inside the \$HOME/.subversion/config file. This allows the use of external programs. It can also be done by the Emacs diff mechanism. Emacs selects the behaviour from the value of the vc-svn-diff-switches user-option. Set that user option to: - Unspecified (nil) : to select the Emacs diff mechanism. - None (t) : to select the tool identified by the \$HOME/.subversion/config file.
PSVN Package	The psvn package expects each directory to have a .svn subdirectory as Subversion was before version 1.7
PSVN Source	2002-2009 @ svn.apache.org 2002-2012 @ Emacs Wiki 2015-07-20, 21:42:00 (but also identified as being from 2015-08-20) @ http://www.xsteve.at/prg/emacs/psvn.el identified in Emacs Wiki Svn Status Mode / Psvn as the latest version