Software Configuration Management at Acme Widget Company - NJ

ClearCase Quick Reference [Unix]

This document is available online via the AWC ClearCase home page, http://devnfs1/dist/ClearCase

#	Command (*or URL*)	Description
A. Setting up your basic environment		

1	`$ type cleartool ||`
	`    PATH=/usr/atria/bin:$PATH`
	`#` *add local scripts to your path:*
	`$ PATH=$PATH:/dist/ClearCase/usr/bin`
		If your machine does not have the ClearCase commands available, set `PATH` yourself. For SunOS, your access is from your personal Sun workstation; for Linux, use the machine `devserv1`. Second directory is for local tools Tim has written.

2	`$ alias ct=cleartool`	No one likes to type a long command name. Aliases work in `ksh` and `bash`.
3	`$ id –a  # newgrp vob_two` *if need be*	Ensure your primary group is `vob_two`; if not, use `newgrp` command to change: `newgrp vob_two` (this invokes a sub-shell). All AWC-NJ ClearCase work requires you to be in the correct Unix group, with `umask 2`. Some projects (like BlackBox) might have their own ClearCase group other than `vob_two`.
4	`$ cleartool` *# with no arguments*	Be aware of `cleartool` with no arguments—you enter a mode in which its commands are entered to it directly and you are not in a shell. This is usually less convenient (no shell command line editing), so `quit`.
B. Getting familiar with views and VOB's		
11	`$ ct lsvob | grep –i vista`	Show (VISTA) VOB's available in your region. Ones marked with `*` are mounted.
12	`$ ct mount /vobs/example`	Ensure the VOB you want to use is mounted by ClearCase on your machine.
13	`$ ct lsview –s | grep $LOGNAME`	See what views are defined for you; most view names contain a user ID and usually a project (like `LOD`) and task, and sometimes a platform suffix (`_NT`).
14	`$ ct setview` *viewname*	Set a view context to the view you want to use. This invokes a sub-shell.
15	`$ cd /vobs/example`	Go to the VOB root directory to access your files.
16	`$ umask 022` # or `002`	To collaborate with colleagues, make sure all files you create are readable by group (a previous step ensures that your primary group is `vob_two`). Or writable?
17	`$ ct pwv`	Show me what view context I am in, if any, i.e. my "current view."
18	`$ ct catcs`	List the *configuration specification* ("cspec") of my view.
C. Working with files in your view—the most common commands		
21	`$ cd` *directory*	Move around anywhere with the `/vobs` tree, or elsewhere.
22	`$ ls –l`	Show the files in `$PWD` using normal Unix command. Look at permissions on the file; they're *not writable or editable until you check them out*.
23	`$ ct checkout –unr foo.c`	Check out `foo.c` for editing; use the `-unr` option if someone else has it reserved.
24	`$ touch temp.c; make`	Create a *view-private* file. Build in a view, creating `.o` view-private files.
25	`$ ct ls –l`	Show the files in `$PWD` and how ClearCase sees them (the version, if private, etc).
26	`$ cleardiff foo.c@@/main/LATEST foo.c` `$ xcleardiff` *<args>* *# for graphical output*	Use command to show differences between files. In this example, show the changes from the most recent checked in file on `/main` branch to my checked out file.
27	`$ ct uncheckout foo.c`	Un-checkout the file if you don't want to change it.
28	`$ ct checkin foo.c`	Check-in file to save your changes. **Check-in files often** (better backups!).

29	`$ ct co . && ct mkelem temp.c &&`
	`  ct ci temp.c  && ct ci .`
		Create a new *file element* (a "file in a VOB") using the *view-private* file `temp.c` that you copied into the directory from somewhere else.

30	`$ ct co . && ct mkdir newdir &&`
	`  ct ci .`
		Create a new *directory element* (a "file in a VOB") to add new files to.

31	`$ ct co . && ct mv foo.c FOO.c &&`
	`  ct ci .`
		Rename a file in VOB—note that you must *checkout* the directory first, since you are modifying the contents of the directory. Then check the directory back in too!

32	`$ ct co . && ct rmname foo.c &&`
	`  ct ci .`
		"Remove" or "delete" a file in VOB. Directory must be checked out, like for `mv`. *Use with caution*.

D. Using the ClearCase online help, tutorial, and printed documentation		
41	`$ ct man [-g]` [*command-name*]	Use (graphical) man page tool. Omit the `-g` option if you want plain text output.
42	`$ ct help` [*command-name*]	List a syntax summary of all `cleartool` commands, or just on one.
43	`$ hyperhelp cc_tut.hlp`	Start the ClearCase graphical tutorial system.
44	`$ ct apropos merge`	Show info about the "merge" topic, like the Unix command `apropos`(1) does.
45	http://devnfs1/dist/ClearCase/Manuals	URL to directory of PDF files of "printed" versions of ClearCase user manuals.

	E. Getting information about the change history of files	
51	`$ ct lsco –l –recurse –user tgill`	List all checkouts in this vob, in entire directory below `$PWD`, for the user `tgill`.
52	`$ ct lsvtree [–g] foo.c`	[Graphically] show the version tree information for the file *element* named `foo.c`.
53	`$ ct lshistory –g`	Show history of activity for current directory, graphically.
54	`$ xclearcase`	General graphical tool for browsing source code and other tasks in ClearCase.
	F. Other useful commands—use `ct man` command for full syntax	
61	`$ ct lstype –kind brtype`	Show the branch type *objects* defined for the current vob.
62	`$ ct lstype –kind lbtype`	Show the label type *objects* defined for the current vob.

63	`$ ct find . –version brtype "(aa)" \`
	`-print`
		List file versions that exist on branch type `aa` for the current directory sub-tree.

	G. Power user mechanisms in ClearCase	

71	`$ cd /vobs/example/foo.c@@ &&`
	`find . –type f &&`
	`find . –type d`
		Change directory to the *root* of the *version tree* for the element `foo.c` in the vob `example`, and then list all the versions for that element, followed by all the branches. This shows the power of *version-extended pathnames*.

72	`$ cd /view &&`
	`diff default/vobs/example/foo.c \`
	`vobadm/vobs/example/foo.c`
		Change directory to the *view root directory* and use that mechanism to easily compare the version of the file `foo.c` in the two different views, `default` and `vobadm`. The views have to be active on your machine for this *view-extended pathname* access to different views to work: use `ct lsview` to check (asterisks show *active* views).

73	`$ cd /view &&`
	`dircmp \`
	`-s –d \`
	`default/vobs/`*<your-vob>* `\`
	<your-view>`/vobs/`*<your-vob>* `\`
	`| uniq`
		A quick way to see which files are different between the work in your view and the `/main/LATEST` code for the vob you are working on. The `-d` option shows the changes (in `diff`(1) format) within files that exist in both views.

	H. Local Acme Widget Company - NJ ClearCase tools	
81	`$ PATH=$PATH:/dist/share/ClearCase/usr/bin`	Make local scripts from the ClearCase administrator available to you.
82	`$ releaseDiff -?`	Local script that shows difference between 2 labeled application releases.
83	`$ applyLabel -?`	Local script that applies a label to a VOB.
84	`$ showMerges`	Local script to quickly and succinctly show merges needed from other views.
84	`$` **`report-CC-problem`** *<failing-command>*	Request help from ClearCase administrator with a command that does not work.

85	`$ . /dist/Cle*/usr/quick-functions;`
	`cc_f`
		Define some `ksh` (Korn shell) functions for quick invocation of common ClearCase commands. Use the function named `cc_f` to show all functions or some subset, based on the optional argument matching a string in the comment for a function.

	I. Merging tools & idioms	
91	`$ clearmrgman`	Manages the merging of versions graphically: easiest interface.
92	`$ xclearcase`	Another way to approach merging graphically
93	`$ xcleardiff`	(Compare and/or) merge files graphically
94	`$ ct lsvtree –g` *<file>*	Use graphical version tree browser to merge a single file.
95	`$ ct merge [-g]` *<args>*	Merge files with the command line. Not recommend for most developers.
96	`$ ct findmerge` *<args>*	Complicated tool to determine which files have to be merged.

97	`$ ct findmerge /vobs/`*<your-vob-dir>* `\`
	`-type d -fversion \`
	<branch_merging_from> `-merge`
		A typical command to merge directories, from some branch *<branch_merging_from>* into your current view. Should be done before merging files, if you have added any files or directories to your vob. An example of the `–fversion` option is `.../nj_bugfix_1.1` to merge the latest code on the a bug fix branch.

98	`$ ct findmerge /vobs/`*<your-vob>* `\`
	`-type f -fversion \`
	<branch_merging_from> `-okgmerge`
		A typical command to merge files, from some branch *<branch_merging_from>* into your current view. The `-okgmerge` option requests that ClearCase attempt automatic merging but requires you to approve each merge before it completes..

	K. Other Helpful ClearCase Documentation	
	Most developers do not seem to like to read manuals, but when they do, they appreciate knowing which ones, or which parts, are the most useful for them. Since there are 4,000+ pages of ClearCase manuals, this section provides links (clickable when this document is browsed as a PDF file in your browser) to the most useful sections of Rational's ClearCase manuals (in my opinion), and other references.	
101	Part of Chapter 1, "Introduction to ClearCase ", from the Rel. 3 (1996) *ClearCase Concepts* manual: 13 pp.	An excellent overview of ClearCase concepts—the best single place to get started. (Too bad Rational removed it from the documentation in later releases.)
102	A public web page from Sweden: 7 pp.	A reasonable, short overview of ClearCase from a user's perspective.
103	§5.1 to §5.3 in Chapter 5, "Working on a Team," in the *Working in Base ClearCase* manual: 18 pp.	A good introduction to working with on a development team using ClearCase, working with branches, and doing merges.
104	Chapter 14 in the *Managing Software Projects* manual (Unix or Windows edition): 12 pp.	A more detailed overview of how merging works and typical merging scenarios.