Transparent
Use of
ClearCase

or

How to obtain the version 19 of stepo.F AL08 in the branch of my neighbor using the modules belonging to the bugfix of CY18T1 a Saturday morning at 6h38.

TUC 4.0 March 1998

SCEM/PREVI/COMPAS/GCO

Contents

1. Basic stuff to forget about ClearCase

Believe it or not, "ClearCase is a comprehensive *software configuration management* system. It manages multiple variants of evolving software systems, tracks which versions were used in software builds, performs builds of individual programs or entire releases according to user-defined version specifications, and enforces site-specific development policies"[1].

Human user should only retain that:

- ClearCase allows each user to develop and modify separatly a shared database of objects (*.F, *.U, *.h, *.o ...) in his own Unix environment.
- Access to this database is performed through a "view" which selects a subset of the whole database according to basic rules (which cycle, which release or which particular development...). Very often a "branch" of development is associated to the selected view so that modifications applied in the view do not affect the "main stream" of the database.
- In a near (?) future, it should be possible also to manage build object (*.o) in the same way than source object, avoiding redundant builds and storages.
- A ClearCase database is called a "VOB" (Versioned Object Base).

2. Basic stuff to remember about TUC tools

The following reasons lead to the necessity of a interface level on top of ClearCase:

- Users are supposed to improve scientific quality of the software application and therefore not to spend too much time learning a huge amount of basic ClearCase commands.
- A minimal normalisation (view and branch names) is required in a multiple-users environment.
- The database administration is easier when the administrator controls the behavior of user's commands.

Therefore:

- Users should never use anything else but TUC commands.
- If one user needs to perform an action not include in the scope of TUC commands it either means that:
  - He did a big mistake and he's in serious trouble. He'd better run to VOB administrator's office or use the SOS facility.
  - There is a lack in TUC commands, and the new action should be included in the TUC package for the benefit of all users.

[1] ClearCase Concepts Manual

3. Meteo France Projects

The following projects are curently administrated as ClearCase VOBs:

Name	Project	1st release	Location	
ald	ALADIN	AL04	andante:	marp001/dev/ald
arp	ARPEGE	CY11	andante:	marp001/dev/arp
bat	BATOR	BA01	andante:	marp001/dev/bat
coh	OBSPROC	CO15	andante:	marp001/dev/coh
hlm	HIRLAM physic	HL00	andante:	marp001/dev/hlm
man	MANDALAY	MA01	andante:	marp001/dev/man
ost	OBSORT	OS16	andante:	marp001/dev/ost
tov	Radiative Scheme	TO15	andante:	marp001/dev/tov
xrd	Auxilary Library	XR00	andante:	marp001/dev/xrd

4. Naming conventions

4.1. Release labels

Release labels are always written using upper cases characters. The release labels of the ARPEGE project have changed since CY14R1 and are now numbered CY–T-.
Example:

CY13R5, CY18T1, AL08

4.2. Views and Branches

View names are built as:

project_logname_RELEASE_branch

where **project** is the short name of the project (3 letters), and **branch** is an user-supplied branch-name extension (for esthetic reasons, it should not be too long) and WITHOUT any special character. The associated branch name is:

logname_RELEASE_branch

Public views (read only) are defined for each cycle release, as well as for potential bug-fix (bf) and operationnal (op) branches. Private users can create public views through the `cc_public` command.

Example:

arp_public_CY18
arp_public_CY18T1_op1
arp_public_CY18T1_mrpm602

5. Calling conventions

5.1. Commands

Each TUC function is prefixed with `cc_`. Arguments are introduced with a one-letter option.

Example:

```
cc_init -p arp
cc_dir -f cnt4.F
```

5.2. Standart options

The following options keep the same meaning for most of the cc-commands:

`-b` :	branch name extension, whith possible secundary branches
`-d` :	absolute or relative pathname to a single directory
`-f` :	absolute or relative pathname to a single file
`-l` :	file containing a list of elements
`-p` :	project name (3 letters)
`-q` :	quiet mode
`-r` :	release label
`-u` :	user logname
`-v` :	flag to specifie the current view (no argument)

The `-p` and `-r` options overwrite the default configuration (Cf 6.2).

6. Getting started

6.1. Shell variables

The following paths should be added to the `PATH` variable:

```
/usr/atria/bin
/u/marp/marp001/ccase/bin/users
```

and the following variables should be defined:

```
export CCHOME=/u/marp/marp001
export CCGROUP=marp
export CCMASTER=marp001
```

Check also that the `LOGNAME` variable is correctly set.

If you intend to create modifications on the source files of one project, ask the system administrator to define "`marp`" as a secundary group of your account.

6.2. Initial configuration

Run the `/u/marp/marp001/ccase/bin/users/cc_init` command to define the initial TUC configuration. An initial project name must be supplied.

The following tree is built under the home directory:

```
. sto/
.   .  view/
. ccase/
.   .  conf/
.   .  .  $LOGNAME.conf
.   .  tmp/
```

Further updates are performed with the `cc_update` command. Unless specified in command line, the release is the latest release of the selected project.

`cc_config` displays a short summary of the current configuration (`-l` option for complete description).

A log file is created each day in the `ccase/tmp` directory. Users should be aware that no automatic cleaning is performed.

Examples:

```
# cc_init -p arp

...
 >  Creating /u/mcyc/mcyc002/sto
 >  Creating /u/mcyc/mcyc002/sto/view
 >  Creating /u/mcyc/mcyc002/ccase
 >  Creating /u/mcyc/mcyc002/ccase/conf
 >  Creating /u/mcyc/mcyc002/ccase/tmp
 >  Modify /u/mcyc/mcyc002/ccase/conf/mcyc002.conf file
 >  Your default clear case config file is now:
CCPROJECT .... arp
CCPROJLIST ... ald arp bat coh hlm man ost tov xrd
CCRELEASE .... CY18T1
CCPATH ....... /u/marp/marp001/dev/arp
CCHOST ....... andante
CCDIRS ....... adiab c9xx canari climate control dfi dia kalman nmi
               obs_preproc ocean parallel phys_dmn phys_ec pp_obs setup
               sinvect transform utility var
CCINCL ....... canari_common common function function_ald namelist
```

```
                parameter pointer interface interface_ald
                ald_inc/common ald_inc/namelist ald_inc/interface
                ald_inc/pointer ald_inc/function ald_inc/parameter
CCVIEW ....... /view
CCVIEWSTORE .. /sto/view
...

# cc_config
ald CY18T1 /u/marp/marp001/dev/arp

# cc_config -p ald
arp AL08 /u/marp/marp001/dev/ald

# cc_update -p ald -r AL08
 >  Modify /u/mcyc/mcyc002/ccase/conf/mcyc002.conf file

# cc_config -l
CCPROJECT .... ald
CCPROJLIST ... ald arp bat coh hlm man ost tov xrd
CCRELEASE .... AL08
CCPATH ....... /u/marp/marp001/dev/ald
CCHOST ....... andante
CCDIRS ....... adiab c9xx control coupling dia harryboy inidata
               parallel setup transform utility var
CCINCL ....... common harry_common namelist interface pointer
               function parameter
CCVIEW ....... /view
CCVIEWSTORE .. /sto/view
```

6.3. Getting a view

`cc_getview` displays public views and views belonging to the user. If the view matching to the default configuration (overwritten by the arguments) doesn't exist, `cc_getview` suggests to create it.
An interactive session is closed with the `.  cc_quit` command (note the dot before the command!).

Example:

```
# pwd
/u/mcyc/mcyc002

# cc_config
arp CY18T1 /u/marp/marp001/dev/arp

# cc_getview -p ald -u public
 >  Available views for project ald and release AL08:
 >     0 : exit
```

```
 >     1 : ald_public_AL08
 >     2 : ald_public_AL08_bf
 >     3 : ald_public_AL08_relief
 >  Enter the selected view: 2
...
ald_public_AL08_bf: pwd
/u/marp/marp001/dev/ald

ald_public_AL08_bf: cc_config
ald AL08 /u/marp/marp001/dev/ald

ald_public_AL08_bf: ls
adiab    control     dia        lost+found  setup       utility
c9xx     coupling    inidata    parallel    transform   var

ald_public_AL08_bf: . cc_quit

# cc_config
arp CY18T1 /u/marp/marp001/dev/arp
```

6.4. Accessing multiple projects and branches through the same view

As a basic starting point, a view is created with a very few components: a main release associated with a given project, a branch in this project for any modification purpose:

```
# cc_view
 >
 >  View : ald_mcyc002_AL08_bof
 >  Path : /u/marp/marp001/dev/ald
 >
 >  Rule  Selection                 Action
 >    1.  CHECKEDOUT
 >    2.  mcyc002_AL08_bof
 >    3.  AL08                      [write to mcyc002_AL08_bof]
 >    4.  MAIN                      [write to mcyc002_AL08_bof]
 >
```

which means that you first view any file already selected for modification ("checked out" from the main data base), a file already modified (and therefore in your private branch), or the current cycle AL08 and otherwise, the lastest version available anywhere on the root branch of the data base (MAIN).

If you wish to modify these selection rules, `cc_view -e` provides a set of sub-commands for this purpose (Cf the reference page). For example, if you want to work on the bugfix of the current project and see the cycle CY18T1 of the ARPEGE project:

```
>
 >  View : ald_mcyc002_AL08_bof
```

```
> Path : /u/marp/marp001/dev/ald
>
> Rule  Selection                 Action
>   1.  CHECKEDOUT
>   2.  mcyc002_AL08_bof
>   3.  AL08                      [write to mcyc002_AL08_bof]
>   4.  MAIN                      [write to mcyc002_AL08_bof]
>
> Mod: bb 2 marp01_AL08_bf
>
> Rule  Selection                 Action
>   1.  CHECKEDOUT
>   2.  mcyc002_AL08_bof
>   3.  marp01_AL08_bf            [write to mcyc002_AL08_bof]
>   4.  AL08                      [write to mcyc002_AL08_bof]
>   5.  MAIN                      [write to mcyc002_AL08_bof]
>
> Mod: r 4 CY18T1
>
> Rule  Selection                 Action
>   1.  CHECKEDOUT
>   2.  mcyc002_AL08_bof
>   3.  marp01_AL08_bf            [write to mcyc002_AL08_bof]
>   4.  AL08                      [write to mcyc002_AL08_bof]
>   5.  CY18T1                    [read-only]
>   6.  MAIN                      [write to mcyc002_AL08_bof]
>
> Mod: ok
```

6.5. Editing a file

First get a view, then edit your file with the `cc_edit` command which invokes your favorite editor as defined in the `EDITOR` shell varaible. When exiting the editor, a new version of the file is created unless you specified the `-k` (keep) option. If you keep the file "out" of the database, you should perform a `cc_put` command one of these days...

Example:

`cc_edit -f fphor12.F`
`cc_edit -f cnt4.F -k` and ... `cc_put -f cnt4.F`

6.6. Expanding and managing dependencies of modules

The `cc_depend` command allows the user to expand a single file, a list of files, a branch (including dependents) or the whole current view (useless except for creating the reference binary). The same operations could be performed without expanding the files (simple source copy through the `-s` option). The `-m` flag produces the prioritary list of modules to compile first. See the reference page for advanced features.

Example:

```
cc_depend -b bof -c  /tmp/bigfile -e DOC
```

6.7. On-line documentation

An on-line documentation of the TUC commands is available through the `cc_help` command. The SOS facility send a message to the VOB administrator who is supposed to answer as fast as possible.

Examples:

```
# cc_help dir
 >  Script cc_dir
 >  -------------
 >
 >  Purpose :  Return unix directory of the specified file
 >  -------    as well as associated project and release if
 >             directory found in global tables otherwise
 >             returns only directory and project in the current view [-v]
 >
 >  Usage :  cc_dir -f filename [-p project] [-r release] [-v]
 >  -----
 >
 >  Parameters :  -f = filename
 >  ----------    -p = project name
 >                -r = release label (useless if -v is active)
 >                -v = flag to check in the current view
 >

# cc_help
 >
 >  On-line documentation of cc tools
 >
 >   0. exit
 >   1. cc_add         2. cc_config      3. cc_copy        4. cc_dep
 >   5. cc_depend      6. cc_diff        7. cc_diffbr      8. cc_dir
 >   9. cc_edit       10. cc_fmerge     11. cc_getview    12. cc_help
 >  13. cc_init       14. cc_klviews    15. cc_list       16. cc_log
 >  17. cc_ls         18. cc_mdiff      19. cc_merge      20. cc_mod
 >  21. cc_popul      22. cc_populdir   23. cc_popvob     24. cc_public
 >  25. cc_put        26. cc_quit       27. cc_restore    28.  cc_rmbranch
 >  29. cc_rmlast     30. cc_rmview     31. cc_tree       32. cc_update
 >  33. cc_versions   34. cc_view
 >
 >  SOS: question or remark to the Gourous
 >
```

```
 >  Enter your choice:
```

6.8. View-private storage

As ClearCase is embedded in the Unix environment the user could perform miscellaneous unix commands involving private file systems (create, delete, move...), even in directories of the database. These private files or directories are attached to the view contents and look like permanent storage.

Example:

```
# cd /u/marp/marp001/dev/arp
# cd ocean
# ls -l
total 98
-r--r--r--   1 marp001    marp          2776 Oct 10 12:56 inicou.F
-r--r--r--   1 marp001    marp          1405 Oct 10 12:56 precv.F
-r--r--r--   1 marp001    marp          1875 Oct 10 12:56 sugco0dm.F
-r--r--r--   1 marp001    marp          3124 Oct 10 12:56 sugco0sm.F
-r--r--r--   1 marp001    marp         16556 Oct 10 12:56 wrcodm.F
-r--r--r--   1 marp001    marp          4853 Oct 10 12:56 wrcoedm.F
-r--r--r--   1 marp001    marp          3737 Oct 10 12:57 wrcoesm.F
-r--r--r--   1 marp001    marp          2106 Oct 10 12:57 wrcofl.F
-r--r--r--   1 marp001    marp          8826 Oct 10 12:57 wrcosm.F

# mkdir tmp
# touch toto
# ls -l
total 100
-r--r--r--   1 marp001    marp          2776 Oct 10 12:56 inicou.F
-r--r--r--   1 marp001    marp          1405 Oct 10 12:56 precv.F
-r--r--r--   1 marp001    marp          1875 Oct 10 12:56 sugco0dm.F
-r--r--r--   1 marp001    marp          3124 Oct 10 12:56 sugco0sm.F
drwxrwxr-x   2 marp001    marp            89 Mar 23 08:25 tmp/
-rw-rw-r--   1 marp001    marp             0 Mar 23 08:26 toto
-r--r--r--   1 marp001    marp         16556 Oct 10 12:56 wrcodm.F
-r--r--r--   1 marp001    marp          4853 Oct 10 12:56 wrcoedm.F
-r--r--r--   1 marp001    marp          3737 Oct 10 12:57 wrcoesm.F
-r--r--r--   1 marp001    marp          2106 Oct 10 12:57 wrcofl.F
-r--r--r--   1 marp001    marp          8826 Oct 10 12:57 wrcosm.F
```

6.9. Absolute Clear Case pathnames

In order to acces (read-only status) any source element, one can use an absolute Clear Case pathname, defined as:

local-pathname@@/main/RELEASE

or

local-pathname@@/main/branch_1/.../branch_n/version

The *version* number can be replaced by the *LATEST* keyword.

Example:

```
more adiab/cpg@@/main/CY18T1
vi stepo@@/main/marp001_CY18T1_bf/LATEST
```

7. Commands summary - version 4.0

Arguments in brackets are optional, arguments separated by a vertical line are mutually exclusive.

cc_add -f filename [**-i** import-file] | **-d** directory

cc_config [**-p** project] [**-l**]

cc_copy -l list | **-b** | **-v -o** outdir

c_depend -f file | **-l** list | **-b** branch [**-o**] | **-a** | **-v**

cc_diff -f filename | **-d** directory | **-l** list [**-h**] [**-w**] [**-x**]

cc_diffbr -r release **-b** branch [**-l** list] **-o** outfile

cc_dir -f filename [**-p** project] [**-r** release] [**-v**]

cc_edit -f filename [**-k**] [**-q**]

cc_fmerge -f filename

-c ccpout | **-d** dirout [**-e** ccpdef] [**-t**] [**-s**] [**-m**]

cc_getview [**-p** project] [**-r** release | "all"] [**-b** branch] [**-j** job] [**-u** user]

cc_help [[**-c**] command]

cc_init -p project

cc_klviews

cc_list [**-b** branch] [**-r** release] [**-u** user] [**-o**]

cc_ls [**-r** | **-d**] [**-vob**] [elements]

cc_mdiff [**-r** release] [**-b** branch] [**-u** user] [**-o** outformat]

cc_merge [**-f** filename | **-d** directory | **-l** list]
-v view-tag | [**-r** release] [**-b** branch] [**-u** logname] **-m** mode [**-x**]

cc_mod -f filename | **-d** directory [**-r** release]

cc_popul -d directory

cc_popvob **-p** project

cc_public [**-b** branch]

cc_put **-f** filename | **-d** directory

. **cc_quit**

cc_rmbranch [**-f** filename | **-d** directory] [**-q**] [**-t**]

cc_rmlast **-f** filename | **-d** directory

cc_rmview [**-p** project] [**-r** release | "all"]

cc_update [**-p** project] [**-r** release]

cc_versions [**-p** project] [**-r** release] [**-a**] [**-b**] [**-u** user]

cc_view [-s] [**-t** view-tag] [-e]

Synopsis

cc_add -f filename [**-i** import] | **-d** directory [**-u**]

Arguments

filename	-	name of the file to be added in the current directory
import	-	import file to be set as the first version of the added file
directory	-	name of the directory to be added in the current directory

Discussion

cc_add creates a file or a directory in the current directory for the current view. A new element is created, and a branch matching to the current view is created on the version 0 of the element.

If the -u flag is set, the checkout and the checkin of the current directory is up to the user.

Example1

```
# cc_add -f newfile.F -i ~/tmp/first.F
....
```

Example2

```
# cc_add -d newdir
....
```

Warning

If the name of the file exists in another view, a symbolic link is created to this file, however the created file is empty (unless the -i option is used).

Synopsis

cc_config [-p `project`] [-l]

Arguments

`project` - project name in 3 letters. By default, the one defined in the LOGNAME.conf file

Discussion

Displays the current configuration of the user or the last one matching to the selected project. `cc_config` doesn't change the default configuration, even if the specified project is different of the default one (this operation is performed through `cc_update`). The **-l** option displays the complete list of parameters as they are set in your current configuration.

Example

```
# cc_config -p arp
arp CY18T1 /u/marp/marp001/dev/arp
```

Synopsis

cc_copy -l `list` **| -b | -v -d** `outdir`

Arguments

`list`	-	name of the file containing the list of elements to extract
`outdir`	-	path of the directory to receive the copies

Discussion

`cc_copy` extracts the specified elements (from a list, a branch or the whole view) from the current database and makes a copy with the same unix structure.

Example

```
# more list_elem
cpg.F
stepo.F
yomdim.h

# cc_copy -l list_elem -d ~/tmp/bof
>  adiab/cpg.F copied
>  control/stepo.F copied
>  common/yomdim.h copied

# ls -R ~/tmp/bof
adiab    common   control

/u/mcyc/mcyc002/tmp/bof/adiab:
cpg.F

/u/mcyc/mcyc002/tmp/bof/common:
yomdim.h

/u/mcyc/mcyc002/tmp/bof/control:
stepo.F
```

Synopsis

cc_depend -f `filename` | **-l** `list` | **-b** `branch` [**-o**] | **-a** | **-v -c** `cppout` |
-d `dirout` [**-t**] [**-s**] [**-m**] [**-e** `cppdef`]

Arguments

`filename`	-	single file name
`list`	-	path to a file containing a list of file names to expand
`branch`	-	comma separated list of branch names (the first one should be the branch name extension while the followings should be complete branch names).
`cppout`	-	file to receive the cpp outputs
`srcout`	-	file to receive the source concatenation
`dirout`	-	directory to receive the cpp outputs
`cppdef`	-	comma-separated list of cpp define options

Discussion

Expands a single file or a set of files as well as their dependencies. The `-v` option performs an expand of the entire view. If the `-b` option is used, `cc_depend` looks for dependencies of the include files modified in the branch. If the `-o` flag is set, then dependencies in other projects are also expanded as long as the release(s) of these external projects are defined in the view selection rules.
The `-a` flag makes you possible to expand all the files from the current project which depend of include files in any of the branches matching the current view selection rules.
The `-s` option do not performs cpp call but only a simple copy of the selected source elements.
The `-m` flag produces a *comp_list* file in the output directory (`-d` option) containing the list of modules to compile first in the rigth order to ensure that module dependencies will be correctly handle.
The `-t` option associated to the `dirout` parameter produces a gziped tar of the directory.

Example1

```
# cc_depend -f cnt4.F -c ~/tmp/cnt4.f -e DOC
...
```

Example2

```
# cat my_list
cnt3.F
cnt4.F
# cc_depend -l mylist -s ~/tmp/myroutines -e DOC
...
```

Example3

```
# cc_depend -b bof -o -d ~/tmp/mybranch_and_dep_other_projs
...
```

Warning

`cc_depend` supports multibranch mechanism as long as these branches are correctly defined in the view's configuration rules.

The `-o` flag is only used with the `-b` option. In any case, check your view's configuration rules to be sure that you can access the rigth stuff. Anyway, `cc_depend` displays the origin of each element expanded.

Only the dependencies which do not belong to one of the selected branches are displayed when running `cc_depend`.

Synopsis

cc_diff -f `filename` | **-d** `directory` | **-l** `list]` [-v] [-h] [-w] [-x]

Arguments

`filename`	-	name of the file to compare
`directory`	-	name of the directory to compare
`list`	-	name of a file containing a list of elements to compare

Discussion

`cc_diff` performs the difference between a specified element (using `-f` or `-d` options) of the current view and the predecessor version. If the `-h` flag is set, the element's history is displayed for an easy choice of versions to compare.
The list option allows a very flexible way to compare elements or external files (not in the database). The elements to compare are described in the list-file with the following format:

element-name RELEASE or
element-name [user_RELEASE_branch[/version — /LATEST]] or
element-name or *pathname (external)*

By default differences are displayed using the Cleardiff graphical interface (the `-v` flag stacks vertically). One can either choose a text difference in a separate window (`-w` option), or a serial text difference(`-x` option)

Example1

```
# cc_diff -f cobs.F -h -x
...
```

Example2

```
# cat list
cobs.F mrpa645_CY13R3_4dinc
~/tmp/my_cobs.F
cobs.F CY13R2
# cc_diff -l list -w
...
```

Synopsis

cc_diffbr -r `release` **-b** `branch` **-o** `output` [**-l** `list`]

Arguments

`release`	-	release name
`branch`	-	complete branch name
`output`	-	output file
`list`	-	file containing list of elements to compare

Discussion

`cc_diffbr` produces a file with the comparaison between a list of elements in the specified branch and the specified release.

Example

```
# cc_diffbr -b marp001_CY18T1_op0 -r CY18 -o ~/tmp/diffop0
./adiab/lavent.F CY18
./adiab/lavent.F marp001_CY18T1_op0
...
# cat ~/tmp/diffop0
############################################################
DIFFERENCE BETWEEN BRANCH marp001_CY18T1_op0 AND RELEASE CY18
############################################################
./adiab/lavent.F CY18
./adiab/lavent.F marp001_CY18T1_op0
********************************
<<< file 1: ./adiab/lavent.F@@/main/CY18
>>> file 2: ./adiab/lavent.F@@/main/marp001_CY18T1_op0/LATEST
********************************
-----[after 106 inserted 107-110]-----
...
```

Synopsis

cc_dir -f `filename` **[-p** `project`**]** **[-r** `release`**]** `[-v]`

Arguments

`filename`	-	name of the unknown file
`project`	-	project name in 3 letters
`release`	-	release label

Discussion

Returns the location of the specified file according to the default project and release or to the one specified in the command line. If the `-v` flag is set, the location is found in the current view, regardless of the release label.

Example

```
# cc_dir -f aplpar.F -r CY14T4
adiab arp CY14T4
# cc_config
arp CY15 /u/marp/marp001/dev/arp
# cc_dir -f aplpar.F
phys_dmn arp CY15
# cc_dir -f foo.F
unknown arp CY15
# cc_dir -f cnt4.F -v
control arp
```

Synopsis

cc_edit -f `filename` [-k] [-q]

Arguments

`filename` - absolute or relative path name

Discussion

Edits the specified file for modifications purpose. The current location doesn't really matter since `cc_edit` will always find the file in the database. The editor defined by the `EDITOR` shell variable is used. When exiting the editor the file goes back to the database as a new version unless the `-k` flag is specified.
The `-q` flag activates the "quiet" mode, ie: other versions of the same file for the same release are not displayed before editing the file (you must use this option in a non-interactive mode).

Example

```
# cc_edit -f cnt4.F -k
# cp ~/tmp/new_cnt4.F cnt4.F
# cc_put -f cnt4.F
```

Warning

You always edit what you see. For example, if you want to work on the basis of a cycle and its bugfix, be sure that the bugfix branch is defined in your view selection rules. In this case a secundary branch will be created for the selected element.

Synopsis

cc_fmerge -f `filename`

Arguments

`filename` - output of a previous `cc_merge` command

Discussion

Interactive merge according to the outputs of a previous `cc_merge -m print` command. Automatic and step by step merges are separatly suggested. This command could be launched several times to complete the merge action.

Example

```
# cc_merge -m print -b bf > ~/tmp/lastmerge
# cc_fmerge -f ~/tmp/lastmerge
 > Here is the list of files to be merged automatically :
./c9xx/ebien.F
./c9xx/eincli1.F
./c9xx/eincli2.F
./c9xx/eincli4.F
./c9xx/eincli5.F
./c9xx/eincli6.F
./c9xx/einter10.F
./setup/suecuv.F
Number of files to be merged automatically: 8
 > Do you want to merge automatically these files [y/n]? n

 > Here is the list of files to be merged by step :
./c9xx/erelspe.F
Number of files to be merged by step: 1
 > Do you want to merge these files [y/n]? y
 > > Do you want to merge ./c9xx/erelspe.F [y/n]? y
...
```

Synopsis

cc_getview `[`**-p** `project] [`**-r** `release | "all" ] [`**-b** `branch] [`**-j** `job] [-`**u** `user]`

Arguments

`project`	-	project name in 3 letters
`release`	-	release label
`branch`	-	comma separated list of branch names (the first one should be the branch name extension while the followings should be complete branch names).
`job`	-	job to execute in a non-interactive session
`user`	-	user logname

Discussion

Displays user's views matching to the arguments. Even if a branch name extension is supplied, the other views for the same project and release are also displayed. The predefined keyword "all" used with the `-r` option lists all the avalaible views belonging to the user.
If a job is supplied through the `-j` option, the matching view (if any) is directly selected without any prompt and the command `job` is executed in the sub-shell of the new selected view. This option allows non-interactive sessions to access the database.

Example1

```
# cc_getview -p arp -r CY15 -u public
 >  Available views for project arp and release CY15:
 >     0 : exit
 >     1 : arp_public_CY15
 >     2 : arp_public_CY15_bf
 >     3 : arp_public_CY15_mval003
 >     4 : arp_public_CY15_bugs
 >     5 : arp_public_CY15_marp001
 >  Enter the selected view: 4
 >
 >  View : arp_public_CY15_bugs
 >  Path : /u/marp/marp001/dev/arp
 >
 >  Rule  Selection                Action
```

```
 >    1.  mrpm603_CY15_bugs        [read-only]
 >    2.  CY15                     [read-only]
...
```

Example2

```
# cc_getview -p arp -r CY15 -b bof,mrpm603_CY15_bugs,marp001_CY15_bf
 >  Available views for project arp and release CY15:
 >     0 : exit
 >     1 : arp_marp001_CY15
 >     2 : arp_marp001_CY15_bf
 >     3 : arp_marp001_CY15_implic
 >     4 : arp_marp001_CY15_tocy15t1
 >     5 : create arp_marp001_CY15_bof
 >  Enter the selected view: 5
...create stuff
 >
 >  View : arp_marp001_CY15_bof
 >  Path : /u/marp/marp001/dev/arp
 >
 >  Rule  Selection                Action
 >    1.  CHECKEDOUT
 >    2.  marp001_CY15_bof
 >    3.  mrpm603_CY15_bugs        [write to marp001_CY15_bof]
 >    4.  marp001_CY15_bf          [write to marp001_CY15_bof]
 >    5.  CY15                     [write to marp001_CY15_bof]
 >    6.  MAIN                     [write to marp001_CY15_bof]
...
```

Example3

```
# more /u/marp/marp001/tmp/job
ls ocean

# chmod 755 /u/marp/marp001/tmp/job

# cc_getview -r CY15T1 -j /u/marp/marp001/tmp/job -u public
inicou.F   sugco0.F   wrco.F     wrcoe.F    wrcofl.F
```

Synopsis

cc_help [[**-c**] command]

Arguments

`command` - TUC command (prefixed by `cc_` or not)

Discussion

Displays on-line documentation for the specified command or lists the available commands. Supplied with the list of commands an "SOS facility" submits a message to the VOB administrator for any remark or question.

Example1

```
# cc_help
 >
 >  On-line documentation of cc tools
 >
 >   0. exit
 >   0. exit
 >   1. cc_add         2. cc_config      3. cc_copy        4. cc_dep
 >   5. cc_depend      6. cc_diff        7. cc_diffbr      8. cc_dir
 >   9. cc_edit       10. cc_fmerge     11. cc_getview    12. cc_help
 >  13. cc_init       14. cc_klviews    15. cc_list       16. cc_log
 >  17. cc_ls         18. cc_mdiff      19. cc_merge      20. cc_mod
 >  21. cc_popul      22. cc_populdir   23. cc_popvob     24. cc_public
 >  25. cc_put        26. cc_quit       27. cc_restore    28. cc_rmbranch
 >  29. cc_rmlast     30. cc_rmview     31. cc_tree       32. cc_update
 >  33. cc_versions   34. cc_view
 >
 >  SOS: question or remark to the Gourous
 >
 >  Enter your choice: ...
```

Example2

```
# cc_help depend
 >  Script cc_depend
 >  ----------------
```

```
> ...
```

Synopsis

cc_init -p project

Arguments

project - initial project name

Discussion

Sets the default configuration for running TUC commands (See the introduction).

Example

```
# cc_init -p arp
 !!! Welcome in the Clear Case world !!!

 ...don't panic, only the ClearCase administrator has the right
    permissions to panic...

 1. paths to be added (in your .login or .profile)
 --------------------

 >  /usr/atria/bin
 >  /u/marp/marp001/ccase/bin/users

etc.
```

Warning

The latest release for the specified project is used by default. Further modifications to the default parameters should be performed using `cc_update`.

Synopsis

cc_klviews

Arguments

`none`

Discussion

`cc_klviews` kills view server processes belonging to the user. These command should be call periodically by the users since view servers remains active as long as they are not explicitly killed (or the machine reboots). At least, when a message containing **`cannot fork`** is displayed after any UNIX command, it's time to try `cc_klviews`!

Example

```
# cc_klviews

# nothing more to say!
```

Synopsis

cc_list `[-b [branch]] [-r release] [-u user] [-o]`

Arguments

`branch`	-	comma separated list of branch names
`release`	-	label release
`user`	-	user logname

Discussion

Displays the contents of the default or specified branch. The first field of the comma separated list should only contain the branch name extension of the major branch. The `-o` flag produces a formatted display.

Example

```
# cc_list -b bf -u marp001
...list the bugfix contents
```

Warning

If the branch does not belong to the current view, the user cannot see new elements but only modified directories.

Synopsis

cc_ls [-r | -d] [-vob] [elements]

Arguments

`elements` - list of elements to describe

Discussion

`cc_ls` performs an usual unix `ls` command but displays also the version seen by the current view. The `-r` option performs a recursive `ls` and `-d` describes the directory itself, not its contents. The `-vob` option restricts the listing to the elements belonging to the current database.

Example

```
# cc_config
arp CY18T3 /u/marp/marp001/dev/arp
# cc_ls -d ocean
ocean                    /main/11
# cc_ls ocean
ocean/inicou.F           /main/4
ocean/precv.F            /main/3
ocean/sugco0dm.F         /main/2
ocean/sugco0sm.F         /main/6
ocean/wrcodm.F           /main/3
ocean/wrcoedm.F          /main/2
ocean/wrcoesm.F          /main/3
ocean/wrcofl.F           /main/4
ocean/wrcosm.F           /main/8
```

Synopsis

cc_mdiff `[-r release] [-b branch] [-u user] [-o outformat]`

Arguments

`release`	-	default release label to compare
`branch`	-	branch name extension to compare
`user`	-	logname of the branch owner
`outformat`	-	format of the output display

Discussion

`cc_mdiff` performs a recursive multi diff between current elements of the view and elements belonging to the specified branch if any, the default release otherwise.
Output format is either **`diff`** (standard diff output), **`lside`** (side-by-side comparaison report style) or **`serial`** (differences are reported as entire lines in above-and-below format instead of side-by-side format).

Example

```
# cc_view -s
xrd_marp001_XR07_xr08
# cd grib
/u/marp/marp001/dev/xrd/grib
# cc_mdiff -r XR07 -o serial
********************************
<<< file 1: ./codega.F@@/main/XR07
>>> file 2: ./codega.F
********************************
-----[234 changed to 234]-----
<  9012                  FORMAT (1H ,'NON-ZERO VALUE IN MISSING DATA',
---
>  9012                  FORMAT (TR1,'NON-ZERO VALUE IN MISSING DATA',
-----[259 changed to 259]-----
<  9000         FORMAT (1H ,'NUMBER OF BITS PER DATA VALUE, ',I3,
---
>  9000         FORMAT (TR1,'NUMBER OF BITS PER DATA VALUE, ',I3,
...
```

Synopsis

cc_merge [**-f** `filename` | **-d** `directory`] [**-r** `release`] [**-b** `branch`] [**-u** `user`] [`-v viewtag`] **-m** `mode` [`-x`]

Arguments

`filename`	-	file name to be merged
`directory`	-	directory to be merged (the directory itself, not its contents) - replaced by the keyword `all` for selecting all the directories which need to be merged
`list`	-	filename including list of elements to be merged
`release`	-	release label of the "from" version
`branch`	-	branch name extension of the "from" version
`user`	-	logname of the owner of the "from" version
`viewtag`	-	complete view tag of the view from which the merge is expected
`mode`	-	executive mode of the merge command (print, auto, step)

Discussion

Performs the merge of a file, a list of files, a directory or an entire branch. The origin of the version to merge is either specified by the association of the command line options `-r, -b, -u` or directly by the `-v` option. By providing a viewtag, the user selects the versions of the elements as they could be seen in the matching view, whatever the view's configuration complexity is.
The `print` mode displays the merges needed but does not perform any action. The `auto` merge performs automatically as often as possible and aborts if a merge is not possible. The `step` mode performs the merges and asks for a confirmation for all the differences.
Basically a merge, is an extension of the "diff" mechanism applied between the two selected versions (the one in the current view, and the other one specified by the arguments) and the common ancestor of these two elements. The `-x` flag disables the graphical interface.

Example1

```
# cc_merge -b bf -u marp001 -m print
 >  Looking for directory merges ...
 >  Merge files ...
Needs Merge "./adiab/cpg.F" [(automatic) to /main/15 from
```

```
/main/marp001_CY15T1_bf/1 (base also /main/15)]
Needs Merge "./adiab/cpglag.F" [(automatic) to /main/11 from
/main/marp001_CY15T1_bf/1 (base also /main/11)]
Needs Merge "./control/cnt4.F" [(automatic) to /main/15 from
/main/marp001_CY15T1_bf/2 (base also /main/15)]
...
```

Example2

```
# cc_merge -v arp_mrpe684_CY15T1_phase -d all -m print
Needs Merge "adiab" [(automatic) to /main/12 from
/main/mrpe684_CY15T1_phase/5 (base also /main/12)]
-----[ added ]-----
> gnhpdvd.F  23-May.08:12 mrpe684
-----[ added ]-----
> gphluv.F  27-Feb.09:42 mrpe686
-----[ added ]-----
> lacone.F  29-Mar.08:52 mrpm603
-----[ added ]-----
> lanhsi.F  23-May.08:21 mrpe684
-----[ added ]-----
> lattey.F  23-May.08:22 mrpe684
```

Warning

Be sure that the environment variable `DISPLAY` is correctly set when using the `step` mode.

`cc_merge` does not support the multibranch mechanism through the `-b` option. One should use the `-v` facility instead.

Synopsis

cc_mod -f `filename | ` **-d** `directory [` **-r** `release]`

Arguments

`filename`	-	name of a file element
`directory`	-	name of a directory element
`release`	-	release label

Discussion

`cc_mod` displays the current versions of the specified file or directory as well as their "real" location (user name and group). The exit status of `cc_mod` is 0 (the element exists without modification), 1 (the element is unknown in the current view) or 2 (the element exists with modifications).

Example

```
# cc_mod -f suct0.F
 >  Already modified versions of suct0.F:
 >  Branch                     User                            Group
 >  mrga568_CY13R3_exergy      MARQUET         Pascal          CNRM/GMGEC
 >  mrpa645_CY13R3_4dinc       THEPAUT         Jean-Noel       GMAP
```

Synopsis

cc_popul -d `directory`

Arguments

`directory` - source directory

Discussion

`cc_popul` updates the data in the current view with the elements of the specified source directory. Does not create any new element. New elements are copied to the `ccase/tmp/nf_`*directory* directory.

Example

```
# cc_popul -d ~/tmp/dir_update
...
```

Synopsis

cc_popvob -p `project`

Arguments

`project` - project name

Discussion

`cc_popvob` is a high level function which populates an entire project with the data of the current directory. It means that after completion, the project is an image of the local directory (including subdirectories). Modification, removing, adding or moving of elements are carried out by `cc_popvob`. For this reason, it should be used carefully.

Example

```
# cc_getview -p arp -r CY18T1
...
# cd $WORKDIR/tmp/arp_new
# ls
adiab          control        namelist       phys_dmn       transform
c9xx           dfi            ...
...
# cc_popvob -p arp
...
```

Warning

Be sure of the correctness of your view configuration rules before running `cc_popvob` and that you are INSIDE the directory location of the new version of the project!

Synopsis

cc_public [**-b** `branch`]

Arguments

`branch` - branch name extension to give to the public view created

Discussion

`cc_public` creates a public view matching to the current one. The name of the public view is either *project_public_RELEASE_logname* or *project_public_RELEASE_branch* as specified in the command line.

Example

```
# cc_view
 >
 >  View : ald_marp001_AL08_bof
 >  Path : /u/marp/marp001/dev/ald
 >
 >  Rule  Selection               Action
 >    1.  CHECKEDOUT
 >    2.  marp001_AL08_bof
 >    3.  marp001_AL08_test       [write to marp001_AL08_bof]
 >    4.  AL08                    [write to marp001_AL08_bof]
 >    5.  CY18T1                  [read-only]
 >    6.  MAIN                    [write to marp001_AL08_bof]
 >
# cc_public -b pubof
 >  Creating the view ald_public_AL08_pubof ...
Created view.
...
```

Warning

Multibranch mechanism and view selection rules edition through the `cc_view -e` command can mostly avoid the need to create a public view.

Synopsis

cc_put -f `filename` | **-d** `directory`

Arguments

`filename`	-	name of the file to replace in the database
`directory`	-	name of the directory to replace in the database

Discussion

`cc_put` replaces in the database an element previously checked out (example: a file edited with the `-k` option of the `cc_edit` command).

Example

```
# cc_put -d ocean
...
```

Warning

Only an existing element could be replaced with `cc_put`. To create a new element, one should used `cc_add`.

Synopsis

. **cc_quit**

Arguments

`none`

Discussion

`cc_quit` exits from the current view and displays a list of checkedout files if any. A dot should be typed before the command to perform the exit of the view in the calling shell.

Example

```
# . cc_quit
```

Warning

The dot preceeding the `cc_quit` command is Korn-Shell specific. This command doesn't work in the C-Shell environment (only type exit).

Synopsis

cc_rmbranch [-f filename | -d directory] [-q] [-t]

Arguments

filename	-	branch-file to remove
directory	-	directory-branch to remove

Discussion

`cc_rmbranch` removes all the data or a particular element (`-f` and `-d` options) belonging to the branch associated to the current view. The branch type itself is not destroyed, so that the view is still valid unless the `-t` flag is set. The quiet mode (`-q` option) suppress the confirmation step.

Example

```
# cc_getview -r CY12 -b test
...
# cc_rmbranch
 >  Contents of the branch marp001_CY12_test
 >  Files:
 >   ./ocean/bid.F
 >   ./ocean/sugco0.F
 >   ./ocean/wrco.F
 >  Directories:
 >   ./ocean
 >  Removing ? [no]
...
```

Warning

This command destroys informations irretrievably...so think twice! Users should be aware that branch mechanism doesn't imply that any extra disk memory cost on their quota since all the deltas (changes from the main cycle) are stored under the `CCMASTER` account. Therefore it could be safer to wait for a general cleaning of branches with backup as they are planned regularly by the VOBs' administrators.

Synopsis

cc_rmlast -f `filename` **|** **-d** `directory`

Arguments

`filename`	-	name of the file whose last version has to be removed
`directory`	-	name of the directory whose last version has to be removed

Discussion

`cc_rmlast` emoves the last version of the specified element belonging to the main user's branch.

Example

```
# cc_rmlast -f sudimo.F
 >  This element doesn't belong to your branch

# cc_rmlast -f adiab/cpg.F
Removing these versions of "adiab/cpg.F":
  /main/marp001_CY15_implic/1
Remove versions?  [no]
....
```

Warning

This command destroys information irretrievably...so keep cool, one more time!

Synopsis

cc_rmview `[-p project] [-r release | "all"]`

Arguments

`project`	-	project name in 3 letters
`release`	-	release label or keyword "all"

Discussion

Removes the selected view. The command behaves as `cc_getview` for selecting the views.

Example

```
# cc_config
arp CY15 /u/marp/marp001/dev/arp
# cc_rmview
 >  Available views for project arp and release CY15:
 >     0 : exit
 >     1 : arp_public_CY15
 >     2 : arp_public_CY15_bf
 >     3 : arp_marp001_CY15
 >     4 : arp_marp001_CY15_bf
 >     5 : arp_marp001_CY15_implic
 >     6 : arp_public_CY15_marp001
 >     7 : arp_marp001_CY15_tocy15t1
 >  Enter the view to remove:
....
```

Warning

Removing a view does NOT remove the contents seen by this view. So that one can re-create later a view to access again this contents. Removing a view slightly increases the file quota.

Synopsis

cc_update [**-p** project] [**-r** release]

Arguments

project	-	project name in 3 letters
release	-	release label

Discussion

Updates the default configuration according to arguments. If no release is supplied, the latest one of the project is selected.

Example

```
# cc_config
arp CY15 /u/marp/marp001/dev/arp
# cc_update -p ald -r AL08
 >  Modify /u/marp/marp001/ccase/conf/marp001.conf file
 >  Your default clear case config file is now:
CCPROJECT .... ald
CCPROJLIST ... aaa ald arp coh hlm pot tov xrd
CCRELEASE .... AL08
CCPATH ....... /u/marp/marp001/dev/ald
CCHOST ....... becarre
CCDIRS ....... adiab c9xx control coupling dia harryboy harry_common
               inidata setup transform utility var
CCINCL ....... common         function   namelist   pointer
               harry_common   interface
CCVIEW ....... /view
CCVIEWSTORE .. /sto/view
```

Warning

`cc_update` changes the general default parameters so that it should not be used inside a view (in this case, only the local default parameters of the view are updated).

Synopsis

cc_versions `[`**-p** `project] [`**-r** `release] [`**-u** `user] [`**-b**`] [`**-a**`]`

Arguments

`project`	-	project name (3 letters)
`release`	-	release label of the project
`user`	-	user logname

Discussion

By default `cc_versions` displays the releases associated with the default or specified project. The `-b` option displays the user's branches, for all releases (default) or for the specified one. The `-a` option overwrites the default user and selects all the users while `-u` selects only one particular user.

Example1

```
# cc_versions -p xrd
 > releases of the xrd project:
 >  > XR00
...
 >  > XR07
 >  > XR07_op1.01
 >  > XR07_op1.02
 >  > XR07_op1.03
 >  > XR08

# cc_versions -p xrd -b -a -r XR07
 >  SEVAULT Eric / SCEM/PREVI/COMPAS (61 07 82 71)
 >  >  marp001_XR07_bf
 >  >  marp001_XR07_op1 active
 >  >  marp001_XR07_xr08 active
 >  ESTRADE Jean-Francois / SCEM/TTI/DEV/VAL (81 38)
 >  >  mval003_XR07_sun active
 >  >  mval003_XR07_val active
```

Warning

The `-a`, `-u` and `-r` options must be used only with the `-b` option.

Synopsis

`cc_view [I-s] [-t viewtag] [-e]`

Arguments

`viewtag` - complete view tag name (prj_logname_RELEASE_ext)

Discussion

`cc_view` returns some informations about the current view (if any) or the specified one (`-t` option): view-name, project path and selection rules. These rules are numbered by order of priority and associated to an "action" which describes what it is possible to perform on an element selected by the current rule. Two predefined rules are very often present at the beginning and the end of the rules: `CHECKEDOUT` which stands for any element with write-permissions in the database (from a `cc_edit -k` for example) and `MAIN` which stands for the last version of any element on the main branch (`/main/`).
The flag `-s` produces a short output (view-name only). The flag `-e` is used to edit and modifiy the view selection rules according to the iterative mechanism [*rules-display/subcommand/rules-display/.../validated-or-undo exit*]. The subcommand synthax is:

`command [rule-number] [arg1] [arg2]`

with the following possibilities:

- `u`: exits without any change (undo)
- `ok`: exits and save changes
- `d n1 [n2] [...]`:
  Suppress the rules with number n1, n2 etc.
- `r n RELEASE`:
  Adds the selection of `RELEASE` after rule `n`.
  This release is read-only.
- `b n branch-name`:
  Adds the selection of branch `branch-name` after rule `n`.
  This branch is read-only.

- `bb n branch-name`:
  Adds the selection of branch `branch-name` after rule `n`.
  Any modification on one element of this branch will be put in the major branch of the view.

- `br n branch-name`:
  Adds both the selection of branch `branch-name` after rule `n` and the selection of the matching release after this new rule.
  Any modification on one element of this release will be put in the branch `branch-name`.

- `bw n br1 br2`:
  Adds the selection of branch `br1` after rule `n`.
  Any modification on one element of branch `br1` will be put in the branch `br2`.

- `bl n branch-label`:
  Adds the selection of CCMASTER's branch with the given branch-label.
  Most of the time, the branch label is of the kind *CYCLE_branch.version* such as `CY18T1_op1.03`.

Example1

```
# cc_view -t ald_marp001_AL08_test
 >
 >  View : ald_marp001_AL08_test
 >  Path : /u/marp/marp001/dev/ald
 >
 >  Rule  Selection                Action
 >    1.  CHECKEDOUT
 >    2.  marp001_AL08_test
 >    3.  AL08                     [write to marp001_AL08_test]
 >    4.  marp001_CY18T1_bf        [read-only]
 >    5.  CY18T1                   [read-only]
 >    6.  MAIN                     [write to marp001_AL08_test]
 >
```

Example2

```
# cc_view -e
 >
 >  View : ald_marp001_AL08_bof
 >  Path : /u/marp/marp001/dev/ald
 >
 >  Rule  Selection                Action
 >    1.  CHECKEDOUT
 >    2.  marp001_AL08_bof
```

```
>    3.  AL08                       [write to marp001_AL08_bof]
>    4.  MAIN                       [write to marp001_AL08_bof]
>
>  Mod: bb 2 marp001_AL08_bf
>
>  Rule  Selection                  Action
>    1.  CHECKEDOUT
>    2.  marp001_AL08_bof
>    3.  marp001_AL08_bf            [write to marp001_AL08_bof]
>    4.  AL08                       [write to marp001_AL08_bof]
>    5.  MAIN                       [write to marp001_AL08_bof]
>
>  Mod: br 4 marp001_CY18T1_test
>
>  Rule  Selection                  Action
>    1.  CHECKEDOUT
>    2.  marp001_AL08_bof
>    3.  marp001_AL08_bf            [write to marp001_AL08_bof]
>    4.  AL08                       [write to marp001_AL08_bof]
>    5.  marp001_CY18T1_test
>    6.  CY18T1                     [write to marp001_CY18T1_test]
>    7.  MAIN                       [write to marp001_AL08_bof]
>
>  Mod: bw 5 marp001_CY18T1_bf marp001_CY18T1_test
>
>  Rule  Selection                  Action
>    1.  CHECKEDOUT
>    2.  marp001_AL08_bof
>    3.  marp001_AL08_bf            [write to marp001_AL08_bof]
>    4.  AL08                       [write to marp001_AL08_bof]
>    5.  marp001_CY18T1_test
>    6.  marp001_CY18T1_bf          [write to marp001_CY18T1_test]
>    7.  CY18T1                     [write to marp001_CY18T1_test]
>    8.  MAIN                       [write to marp001_AL08_bof]
>
>  Mod: ok
```