NAME

c4 — CVS like front end to the Perforce SCM system

SYNOPSIS

c4 *p4−command* . . .
c4 update|scan|import [**−acdnvxD**]

DESCRIPTION

C4 provides a CVS like *feel* to Perforce's SCM system **p4**. **C4** is not a substitute for CVS, in that it does not try to mimic CVS's command set.

Like CVS, every file in a user's work-space (*client*) is writable, and can be modified on an ad hoc basis. **C4** provides commands that search a client, looking for files that have been changed, and runs a "*p4 edit*" command on them. Any files that are currently opened for edit, and are no longer different than their repository (*depot*) version, a "*p4 revert*" command is run on them. Files that are missing from the client are refreshed ("*p4 refresh*"). Files that exist in the client that have no matching depot version are reported, to remind the user to add them, and can optionally be added automatically. Files that are out of date with respect to the depot are updated ("*p4 get*").

C4 also provides support for clients without the use of environment variables. Each time **c4** is executed, it searches for a "**.c4**" file, and incorporates its contents into the environment of any *p4* command that it invokes. The "**.c4**" file is usually kept in the root directory of a client.

C4 should be used with Perforce version 97.3 or later. **C4** will work with older versions, but files will not remain writable, reducing the usefulness of **c4**.

CLIENT CREATION

Making a new client for use with **c4** is slightly different than when using **p4** directly. Here are the steps that should be followed:

1. Create a client directory.
2. Create a file called "**.c4**" in the client directory with the following contents:
   P4PORT=<*p4 server port address*>
   P4CLIENT=<*client name*>
3. Change directory to the client directory, and run the command "**c4 client**" to create the Perforce client. Set up the client as usual. Modify the "**Options**" line to enable "**clobber**" and add the option "**allwrite**". For example:
   Options: nomodtime clobber allwrite
4. Populate the client with the command "**c4 update**". Note that "**c4 update**" should be used in preference to "**c4 get**"; with the *clobber* flag set, a modified file may be overwritten, destroying any changes made to it locally.

COMMANDS

If **c4** doesn't recognize a command, it just passes the command and the remaining arguments to **p4**, after incorporating the contents of the "**.c4**" file into the environment. If **c4** recognizes the command, it is executed locally, usually invoking several **p4** commands. **C4** adds three new commands:

c4 update

"**c4 update**" resembles the CVS *update* command. It performs the following actions:

1. Recursively scan the current directory and any sub-directories, gathering information about files, including whether the file has been modified with respect to its depot version.
2. Report any files that are unknown, to remind the user that they should be added. This step can be omitted with the **−x** option. Files which are *ignored* are not reported (see the section "IGNORING FILES" below).
3. Files that are not opened and are different than their respective depot versions are opened with a "**p4 edit**" command. Files that are opened and are now the same as their respective depot version are reverted with a "**p4 revert**" command. Files that are missing from the client are re-

freshed with a "**p4 refresh**" command.

4. Files that are out of date with respect to the depot are retrieved with the command "**p4 get**".

Note that a "**c4 update**" can never overwrite a file because a modified file is always opened for edit with a "**p4 edit**" command before it is retrieved with a "**p4 get**" command. If a modified file is out of date, then the user will need to resolve the differences with a "**c4 resolve**" command before submitting.

c4 scan

"**c4 scan**" is the same as a "**c4 update**, but it does not update out of date files ("**p4 get**").

c4 import

"**c4 import**" is used to import or re-import whole source trees into Perforce. **C4** will figure out all changes that have happened in the source tree and issue "**p4 add**", "**p4 edit**" and "**p4 delete**" commands as appropriate. Apart from files that are added, modified or deleted, **c4** can also handle files that have changed type — a file that was a regular file that is now a directory or symbolic link, or any other combination of change.

The general procedure for import is:

1. Make sure that the tree is up-to-date by doing a "**c4 update**" or a "**c4 sync ...**". This is obviously not necessary if this is a new tree.
2. Replace the tree with the new tree. This almost always means that you should remove the old tree and copy in the new tree, ensuring that files that no longer exist get deleted.
3. Change directory to the root of the tree and run "**c4 import**". This will issue add, edit and delete operations as appropriate.
4. Submit the resulting changes ("**c4 submit**").

Import can be used when moving source from another source control system to Perforce. A lump of source that can be identified as a single change is extracted from the other source control system and submitted using the above procedure. The date of the change can even be set correctly by using the "**p4 change -f**" command after the submit.

.C4 FILE FORMAT

Each time **c4** is executed, it searches for a "**.c4**" file, and incorporates its contents into the environment of any *p4* command that it invokes. The "**.c4**" file is usually kept in the root directory of a client. If a "**.c4**" file is not found, then the correct **p4** environment variables must be set (usually **P4PORT** and **P4CLIENT**).

Every line that begins with an alphabetic character (‘A’ – ‘Z’ or ‘a’ – ‘z’), and contains an ‘**=**’ is added to the environment. Every line that begins with a ‘**:**’ is used as an ignore specification (see the section "IGNORING FILES" below). Every line that starts with a ‘**#**’ character, is empty, or contains only spaces is ignored. All other lines are currently ignored, but may not be in future versions of **c4**.

IGNORING FILES

While scanning, if files are found that are not known to the depot, they are examined to see if they should be ignored. This is done by matching the basename (the last name component) of the file to the ignore list for the current directory and the global ignore list. The ignore list for each directory is stored in a file called "**.c4ignore**" in that directory. The global ignore list is compiled into **c4**, and can be modified or replaced by entries in the **.c4** file. Directories are matched with ignore lists, allowing whole directories to be ignored.

Lines beginning with ‘**:**’ in the **.c4** file are considered ignore specifications. An ignore specification is a globbing string as used by the shell; a file will be ignored if it matches any entry in an ignore list. A ‘**:**’ on a line by itself will cause the entire ignore list to be deleted; this is generally used to clear the global list and replace it with new entries.

The default compiled in global ignore list contains the following:

*.o, *.a, *.Z, *.gz, .c4, tags, TAGS, core.

Here is an example **.c4** file that defines typical environment variables and replaces the builtin ignore list with another:

```
P4PORT=bigsun:1666
P4CLIENT=myclient
:
:*.bz2
:*.[is]
:x.*
:core
```

PERFORCE SUPPORT

C4 uses two as yet undocumented features included in 97.3 and later releases of **p4**:

allwrite

The *allwrite* client option always creates files that are writable in the client. While **c4** does not rely on this, much of its usefulness is diminished without it.

diff -sc

"**c4 diff -sc**" combines the effect of "**c4 diff -sa**" and "**c4 diff -se**", to improve the performance of scanning for changed files. **C4** will only use this feature on versions 97.3 and later. [Note: In the 97.3 release, this feature did not work correctly, so **c4** does not use it. It was later discovered that this feature would not significantly increase the performance of **c4**.]

INSTALLATION

C4 is a single executable that is usually stored in the same place as the **p4** client executable. **C4** may be configured as a replacement front end for **p4** by creating a shell script called **p4**, which contains the following:

```
#!/bin/sh
export P4COMMAND=<name of real p4 command>
c4 "$@"
```

OPTIONS

When one of the local **c4** commands is invoked, the following options may be used to modify its behavior:

−a Force automatic add. When used with "**c4 update**" or "**c4 scan**", unknown files will be added rather than just reported.

−c Canonical update processing. **C4** will run commands necessary to change the depot to reflect the current client, regardless of the changes. [Depreciated; use "**c4 import**".]

−d Force automatic delete. When a file is missing in the client, it will be deleted from the depot, rather than refreshed.

−n Any commands that cause changes will not be executed. When **c4** is invoked with the −n and −d options, the user can see what will be executed for a given circumstance.

−v Verbose. Print more messages about what is happening. This is useful when running **c4** at the root of a large client directory structure; it will print the name of each directory as it is scanned.

−x Messages about unknown files usually reported by "**c4 update**" and "**c4 scan**" are suppressed.

−D When given, **c4** prints diagnostic messages about what is going on (used for debugging **c4**). Multiple −D options can be given to get more details.

SEE ALSO

p4(1), "Perforce User Manual".

BUGS

If you modify a file locally that has been changed in the depot, then when you do a "**c4 update**", you will get a merge conflict. **C4** runs a "**p4 edit**" command followed by a "**p4 get**" command on the file, which will result in messages that may confuse some users; **c4** does the right thing though. You will have to run a "**c4 resolve**" to deal with the merge issues before you can submit the file. Do not run "**c4 sync**" or "**c4 get**" as the messages suggest.

With both "**clobber**" and "**allwrite**" options set on the client (as they should be when using **c4**), you should not use "**c4 get**" to update files from the depot if you have made local changes — your changes will be overwritten. Use "**c4 update**".

If you run **c4** in a directory that is empty in the depot, such as a directory with only new files in it, you may see messages like ". . . – no such file(s)". Don't worry about them.

When doing a "**c4 import**", the client must be up-to-date with respect to the depot before changes are made to the client; that is, there must not be files that must be synchronized in a directory that no longer exists. This is because the **p4** client command does not handle directories; if a directory is replaced by a file or a symbolic link, **p4** can get confused and **c4**'s import processing will fail.

When doing a "**c4 import**", changes between a regular file, directory or symbolic link are handled, however, this functionality is not well tested. This will no doubt be taken care of by the next release of **c4**.

The options provided by **c4** are not necessarily consistent with each other. This is a historical accident and may be rectified in the future.