

Clear Case 开发人员快速参考

新大陆 陆向东

修订历史

| Version | Primary Author(s) | Description of Version | Date Completed |
|---------|-------------------|---|----------------|
| Draft | 陆向东 | 草稿 | 2003-12-25 |
| v0.50 | 陆向东 | 修改 12, 增加 unco -rm xx.cpp 插入 25,26, 创建 branch 和 merge branch | 2003-12-25 |
| v0.51 | 陆向东 | 修改 23, 指明可以为单个目录设置版本号。 | 2003-12-29 |

1、clearcase、VOB、view、config spec、DO 、cleartool,etc.

clearcase 是 Rational 公司发行的配置管理工具, 和 VSS、CVS 等一样应用广泛。在 clear case 中, 所有文件、版本、分支、标签等都纳入 VOB 管理 (versioned object bases), VOB 就像一个大桶一样。每个人通过各自的视图去看这个 VOB, 可以看到不同的代码版本。通过 view 看 VOB 的规则可以配置, 称为 config spec。每个人都可以修改自己的 config spec 去查看你想要看到的代码版本。view 也有自己的私有存储空间, 你自己创建的临时文件, 就存放在自己的 view 下面。DO 是 derived object 的缩写, 就是编译过程中产生的*.o 和*.a 等文件。如果你是采用 make 构造, 则这些*.o,*.a 文件不算 DO, 只是 VIEW 下的私有文件。采用 clearmake 产生的文件才能算 DO。

cleartool 是 clearcase 中最常用的工具, 可以交互使用, 也可以单个命令行执行, 如:

```
$cleartool ls
```

或者

```
$cleartool
```

```
cleartool> ls
```

如果你不能使用 cleartool, 请在 PATH 中增加/usr/atria/bin 路径。

在 cleartool 中可以输入!切换到 shell 中。

2、clearcase 没有中文材料, 怎样查看命令帮助?

```
$cleartool man xxx
```

```
$cleartool help xxx
```

xxx 为命令名字。

3、使用 clearcase 后的工作流程有什么变化?

没有太大变化, 只是工作目录变化了。工作目录在/view/xxx/home/vobs/cc_custcare

如 cd /view/sunbj/home/vobs/cc_custcare

```
$cleartool setview sunbj
```

修改文件前, 先 checkout, 如:

```
$cleartool co -nc XX.cpp
```

修改文件，和以往相同

单元测试，也和以前相同

提交集成，即 check in 如：

```
$cleartool ci -nc XX.cpp
```

在提交前，别人无法看到你的修改。

提交后，到 custcare 视图编译集成。

```
cd /view/custcare/....
```

```
$cleartool setview custcare
```

然后执行 make...

注意，clearcase 提供了构造工具，称为 clearmake，如果能用起来，最好用 clearmake。用 clearmake 构造对象，可以让 DO 在各个视图间共享，可以节省空间。相关命令：clearmake、winkin、view_scrubber、scrubber 等。

另外，现在正在用的 makefile 要修改，请注意 custcare 和其他 view 目录下 makefile 中的路径。

对 custcare 下的代码的 make，应当指定专人。

客服组每个人的 view 所在：

```
/view/wangqj/home/vobs/cc_custcare
```

```
/view/luols/home/vobs/cc_custcare
```

```
/view/sunbj/home/vobs/cc_custcare
```

```
/view/yangj/home/vobs/cc_custcare
```

```
/view/cheny/home/vobs/cc_custcare
```

请大家进去试用，下周就变成正式环境了。

如果最后所有人都到/view/custcare 下直接修改，则还会发生互相覆盖的问题。所有人都应该在自己的 view 下面开发，到 custcare 下集成。

注意事项：开始使用时，请注意产生的*.a，*.so 等别人能不能看到，看到的版本对不对。如何才能让别人使用正确的版本呢？这是 custcare 组集成人员要考虑的问题。--实在不行就 check in 呗。

4、为何输入命令没有反应？

可能是在提示你输入注释。

```
cleartool> mkelem t2.txt
```

```
Creation comments for "t2.txt":
```

```
adfasdfasfsafdsafsaf fdsafdafd
```

5、所谓的版本是什么东东？

在 clearcase 下，每个元素（element）checkout 再 check in，版本就会加 1。如：

```
cleartool> lshistory t.txt
```

```
24-Dec.13:05    luols        create version "t.txt@@/main/3"
```

```
24-Dec.11:53    sunbj        create version "t.txt@@/main/2"
```

```
24-Dec.11:49    custcare     create version "t.txt@@/main/1"
```

```
"create by custcare."
```

```
24-Dec.11:48    custcare     create version "t.txt@@/main/0"
```

```
24-Dec.11:48    custcare    create branch "t.txt@@/main"
```

```
24-Dec.11:48    custcare    create file element "t.txt@@"
```

```
"create by custcare."
```

而下文所说的 V100.000.200 是该版本的标签。如给 t.txt 版本 3 加上 V100.000.2000 的标签。
通过版本和版本标签，都可以引用该元素的正确版本。

6、你现在工作在哪个视图下？

```
$cleartool pwv
```

7、当前工作目录呢？

```
$cleartool pwd
```

8、如何看看你修改的文件的历史？

```
$cleartool lshistory xx.cpp
```

9、如何看看你修改的文件的版本树？

```
$cleartool lsvtree xx.cpp
```

10、如何增加新的目录？

```
$cleartool mkdir dir_name
```

11、如何增加新的文件？

```
$cleartool mkelem file.cpp
```

12、你不想 checkout 某个文件了，在怎么办？

```
$cleartool unco -keep file.cpp
```

在你的当前目录下会产生 file.keep，保留你的修改。

```
cleartool> unco -rm *.pc
```

把你 checkout 的文件删除，你看到的文件都是 check out 前的。

13、为何不让你增加新的文件或者目录？

首先确认，你是在你的 view 下面。

在 clearcase 中，目录也纳入版本管理，在当前目录下创建子目录和文件，要把当前目录也 checkout。如：

```
$cleartool co -nc .
```

. 表示当前目录

14、谁把你想修改的文件 checkout 了？

```
$cleartool lscheckout xx.cpp
```

15、如何查看哪些是 DO？

```
$cleartool lsdo
```

16、 你的 config spec 是怎样的？

```
$cleartool catcs
```

你可以看到如下：

```
element * CHECKEDOUT
```

```
element * /main/LATEST
```

表明你的 view 看到的文件是：首先是所有 checkout 的文件，然后是其他最新的文件。

如果改成：

```
element * CHECKEDOUT
```

```
element * /main/V100.000.001
```

，则你的视图看到的文件是：首先是所有 checkout 的文件，然后是其他 V100.000.001 版本的文件。

如果改成：

```
cleartool> catcs
```

```
element * CHECKEDOUT
```

```
#element * /main/LATEST
```

```
element t/* /main/2
```

```
element * /main/LATEST
```

```
cleartool>
```

则其中 t 目录下的文件取版本 2，其他同上。

请参考 `$cleartool man config_spec`，可以看到更多游戏规则，一般我们用到以上几种就够了。顶多再增加一个 branch。

17、 修改你的 config spec？

```
$cleartool edcs
```

就像用 vi

18、 自己总共 checkout 了多少文件？

```
$cleartool lscheckout -cview -me -avobs
```

19、 如何比较两个文件的不同？

```
$cleartool diff prog.c prog.c@@/main/4
```

比较结果的具体解释见帮助 diff。

20、 如何查看有几个人在争用 clearcase license？

```
clearlicense
```

21、 license 不够时，临时释放自己的 license？

```
clearlicense -release
```

clearcase 不允许一段时间内过多使用这个命令。

22、 哪些命令要慎用？

所有 rm 打头的名字。另外还有 mv 等命令。搞清楚 rmname 和 mv、rmelem 不同含义之后再

23、 每周一次发布都要形成一个版本，如何操作？

在根目录，`cd /view/custcare/home/vobs/cc_sucstcare`

```
$cleartool> mklbtype -nc V001.000.200
```

Created label type "V001.000.200".

```
$cleartool> mklabel -r V001.000.200 .
```

其中. 为当前目录，每周增加一些版号。

以后可以根据版本找到当时发布的代码。

--客服组应该有专人负责这件事情。

可以为单个子目录设置版本标签。

```
$cleartool> mklbtype -nc V001.000.300
```

```
$mklabel -r V001.000.300 ./src/back/wlan
```

24、 两个人不可避免地要修改同一个文件，该如何处理？

最简单的方法，就是要求前一个人赶快 check in。

如果两个人修改文件的历时都很长，都不想 check in，则还有两个办法，

第一个办法：第二个人做 unreserved checkout，然后各自修改，依次提交。

第二个办法，将该文件创建一个 branch，然后第二个人工作在这个 branch 上。需要时作 merge。

参考 checkout | co [-reserved] [-unreserved]，mkbranch、merge 等。

参考下面两个问题。

25、 如何创建一个 branch？

[sunbj view]

先创建 branchtype

```
cleartool> mkbtype -c "test branch" fix_bug11
```

修改 config spec

```
cleartool> edcs
```

```
element * CHECKEDOUT
```

```
element * ../fix_bug11/LATEST
```

```
element bill_print_count_pc.pc /main/LATEST -mkbranch fix_bug11
```

```
element * /main/LATEST
```

```
cleartool> co -nc bill_print_count_pc.pc
```

Created branch "fix_bug11" from "bill_print_count_pc.pc" version "/main/1".

Checked out "bill_print_count_pc.pc" from version "/main/fix_bug11/0".

接下去修改的 bill_print_count_pc.pc 是另外一个分支，不会和别人的修改的冲突。

经过测试后 check in。

26、 如何合并一个 branch？

接下去 bill_print_count_pc.pc 从/main/1 变成了 /main/2

另外一个版本 bill_print_count_pc.pc 的从/fix_bug11/0 变成了/main/fix_bug11/1

其中/main/2 的 bill_print_count_pc.pc 实现了新业务，而/main/fix_bug11/1 的版本修改了一个缺陷。现在要对两个版本合并。

在 custcare 下面 checkout bill_print_count_pc.pc

```
cleartool> merge -to bill_print_count_pc.pc -version /main/fix_bug11/1
```

打开 bill_print_count_pc.pc 文件看看，果然没错。

check in bill_print_count_pc.pc 即可。

27、 如果你能打开 x-server，则还可以用图形工具，如？

/usr/atria/bin/xclearcase /usr/atria/bin/xcleardiff /usr/atria/bin/xlsvtree 等。看看版本树，真是一目了然。

28、 物理的 view 和 vob 在哪里？

```
$cleartool lsvob
```

```
$cleartool lsview
```

千万不要修改 custcare.vbs 和 custcare.vws 及其子目录的任何属性。

总而言之，clearcase 的使用是相当简单的。

还有什么问题吗？有问题马上问。