# Introduction to ClearCase II

# v 1.2

**Vít Hrachový**

**Prague, 26 September 2007**

# Running X Applications

## Running Remote Applications

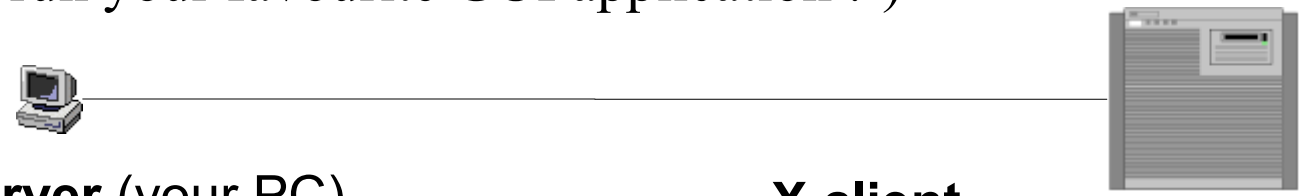
1) Start X server (if needed)

2) get your local IP address

3) log in to the remote machine - via: telnet, ssh, rsh ...

4) set the DISPLAY variable

5) run your favourite GUI application :-)



**X server** (your PC)
- start X (Xfre86e, Reflection)
- get IP (/sbin/ifconfig, ipconfig)

**X client**
set display to your IP, e.g.
- export DISPLAY=**<IP_ADDRESS>**:0

# X settings

## X Server settings

- edit ~/.Xdefaults  to suit your needs (colors, fonts)
- create script to automatically update ~/.display when you boot/login to your PC

## Reflection

- set it to run  /usr/local/bin/xterm
- and on UNIX: export COLORTERM=yes

- set keys:  BackSpace  to generate ^H
  Delete  to generate ^[[3~  (or ^?)

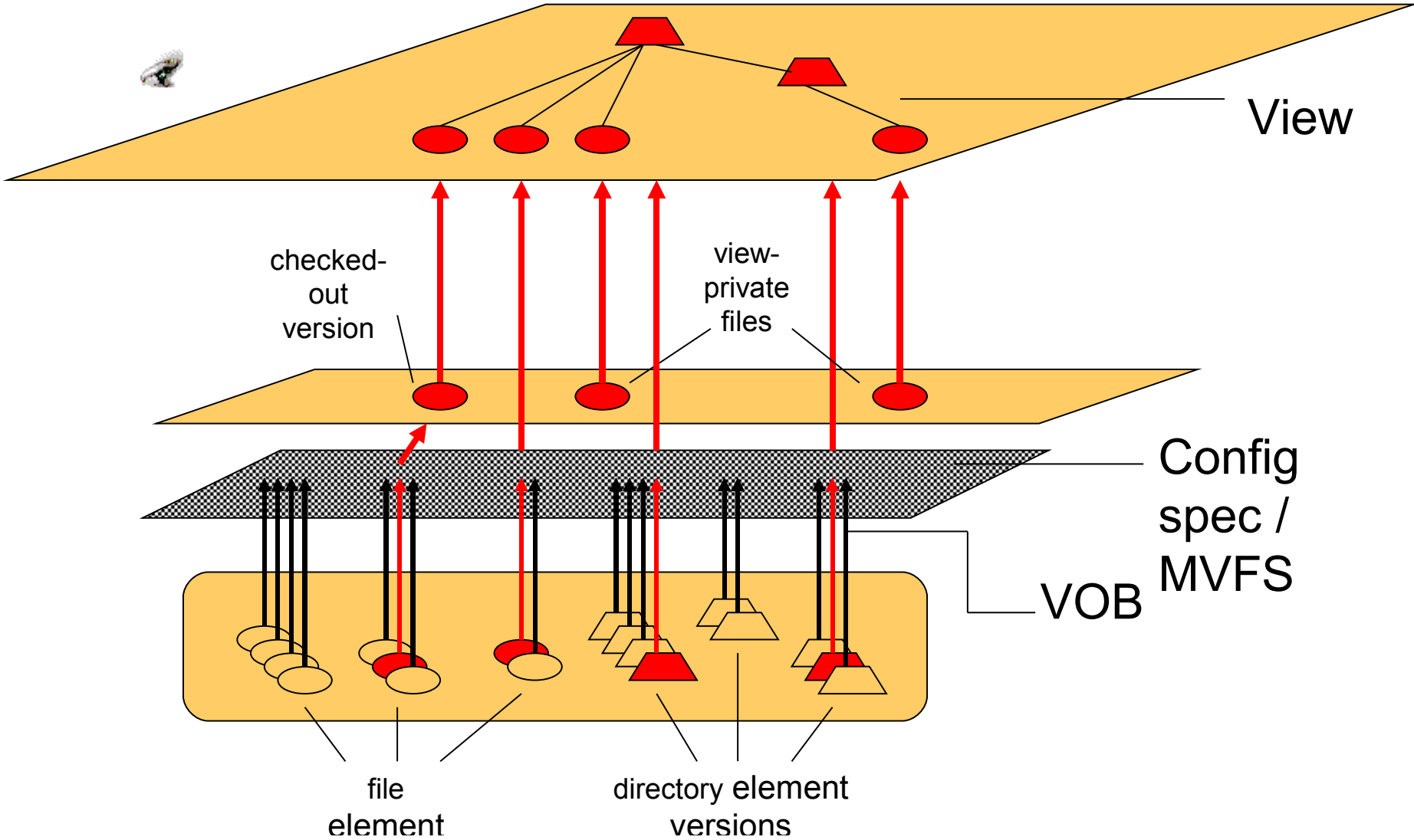
# Working with ClearCase CLI

1. **Log in to the ClearCase host**
2. **(if not automated by your rc scripts)**
   **Set your DISPLAY, add PATH /usr/atria/bin, create the 'ct' shortcut,** e.g.:
   - bash
   - export DISPLAY=158.234.174.114:0
   - export PATH=$HOME/bin:/usr/atria/bin:$PATH
   - mkdir -p  ~/bin
   - ln -s /usr/atria/bin/cleartool ~/bin/ct
3. **(if not automated by your rc scripts)**
   **Set your view**
   - ct setview VIEW_NAME  # the same as our login name

# ClearCase - Getting Help

1. **GUI Help**
   - **xclearcase/ClearCase Explorer/ClearCase Home**
     - menu Help (requires web browser, same source as 3.)
2. **CLI Help**
   - **ct** <*command*> ***-h*** # short online help
   - **ct man** <*command / or something:-)*>
     - (e.g.: cleartool ci config_spec findmerge mkbrtype)
3. **WWW**
   - Documentation is available in form of web pages

# View/VOB concept



# Creating new Element

1. **Ensure, that your config-spec is correct**
   - ct setcs -default  # set the default - only for Training purposes
   - ct catcs  # print current CS
2. **Check out the directory**
   - cd /data/ccase/training/`whoami`/
   - ct co -c 'adding new index.html' **.**
3. **Make an element, edit it**
   - ct mkelem -nc  index.html
   - vi index.html
4. **Check in the directory and the element**
   - ct lsco  # list the checked-out files/directories )
   - ct  ci  **.**  index.html

# Removing Element

- ct rmname – removes element references from actual directory version
  - Check out the directory
    - cd /vobs/training/`whoami`/
    - ct co -c 'CR12345 – remove obsolete files' .
  - Remove the files, links, directories
    - ct rmname -nc obsolete-file obsolete_link obsoleteDir
  - Check in the directory
    - ct ci -nc .
- ct rmelem – destroys complete element with all its history
  - DON'T use rmelem at all, unless you really know what you're doing!!!

# Information about Elements

1. **Information about the current version of a file**
   - ct describe index.html
   - ct describe -g index.html # GUI version - most commands accept -g
2. **History of an element**
   - ct lshist index.html
3. **Version tree**
   - ct lsvtree [ -g ] index.html
4. **What is the current version?**
   - ct ls
   - ct ls -d /data/ccase/training # -d = directory itself, not contents

# Test 1

1. **Change the index.html file. Add your name to it.**

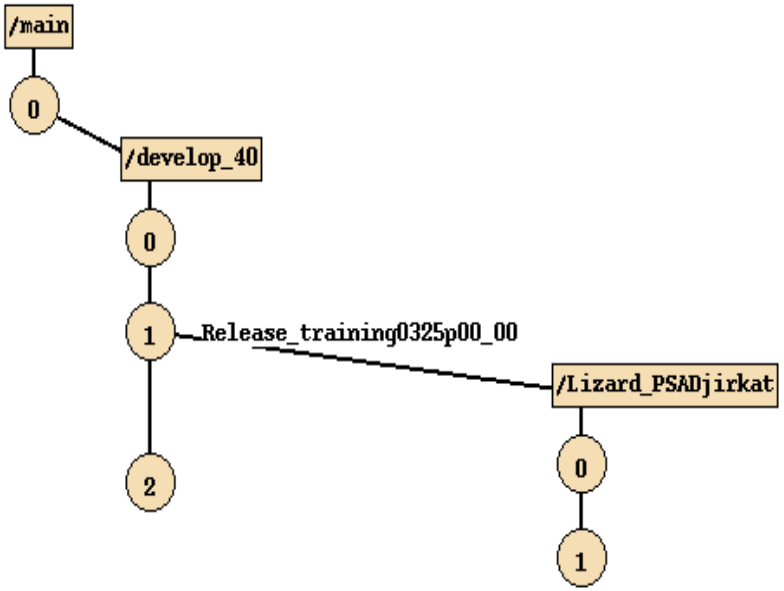
- Tip:
  - checkout the file
  - edit it
  - check it in

```
ct co -nc index.html
vi index.html
ct ci -c 'name added' index.html

( ct describe index.html )
```

# Branches

## Branches are used for:




1. The default stream for a product / component (e.g. “HAB_0.1”); - stable branch

2. Customer specific files, different languages, etc. (e.g. “HAB_0.1_German”)

3. Task branches - to be merged to the default stream as soon as they are stable. (e.g. “XSLT_fix_HABD0123456” or “Lizard_HABD0123456”) - Remedy ticket no.

4. Bug fixes of an old version (e.g. “Rel_1.0_Fixes”, with Lizard_HABD012346 sub-branches)

# Creating Branches

1. **Create a branch type**
   - ct mkbrtype BRANCH_NAME
2. **Set your CS, so you can see the branch (latest version on it)**
3. **and if wanted - set your CS, so the branch is automatically created during a check-out**
   - ct edcs # edit config spec

```
element * CHECKEDOUT
element * .../BRANCH_NAME/LATEST
element * /main/LATEST -mkbranch BRANCH_NAME
```

4. **The branch on a specific file will be created when you**
   - ct mkbranch -nc BRANCH_NAME index.html # or
   - ct co index.html # or
   - ct mkelem new_file

# Test 2

1. **Create a German version of index.html on a branch "HAB_0.1_German" and**
2. **Create a new file "default.css" (on the same branch)**

```
ct mkbrtype HAB_0.1_German
ct edcs     # see below
ct co -nc index.html
vi index.html
ct ci -nc index.html

ct co .
ct mkelem default.css
vi default.css
ct ci -nc . default.css
```

```
element * CHECKEDOUT
element * .../HAB_0.1_German/LATEST
element * /main/LATEST -mkbranch HAB_0.1_German
```

# Test 3

1. **Set config. spec to show the latest version of the main branch.**
2. **List the directory contents**
3. **Where did the 'default.css' file go ?**
   - Tip:
     - save your current config spec to a file:
       ct catcs **>** ~/branch_config_spec.cs
       (you can reuse it later by:  ct setcs FILE)

```
ct edcs     # edit CS
ls
#or: ct setcs -default
#the file is in a different
version of the current
directory
```

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

# Merge

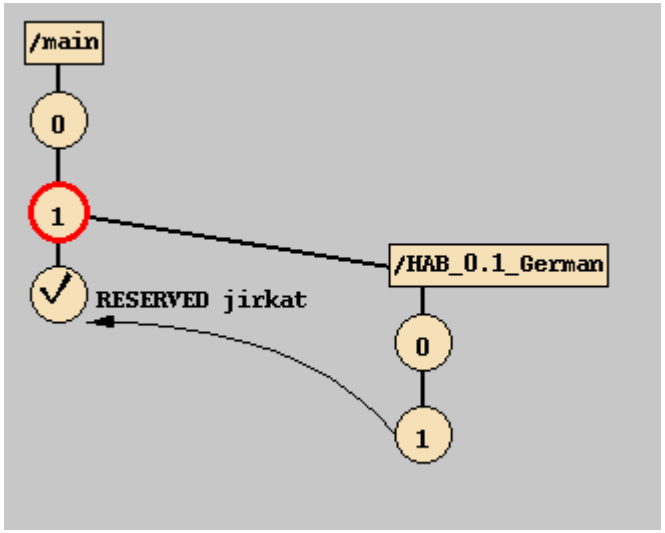
- **Merging changes**
  - set config spec to the **target** version
    **ct edcs**

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

  - create a batch file
    **ct findmerge . -fver .../Lizard_PSADjirkat/LATEST -print**
  - or run graphical merge
    **ct findmerge . -fver .../Lizard_PSADjirkat/LATEST -gmerge**
  - or run merge manually
    **ct co best.c**
    **ct merge -g -to index.html **
    **index.html@@/main/HAB_0.1_German/LATEST**

# Merge

- **Merge result**
  - check in
    **ct ci index.html**



# Labels

**Label is a symbolic name of a certain version of an element**

- There must be a label type defined, before the label can be used.

```
# make a label type
```

- ct mklbtype Stable_1

```
# place it on the current versions of the elements
```

- ct mklabel Stable_1 default.css index.html .

```
# look at the version tree
```

- ct lsvtree -g index.html

# Labels

**In config. spec, you can refer to a version using a label**

```
element * CHECKEDOUT
# for all elements under /vobs/cm/training
# show the version labeled "Stable_1"
element /vobs/cm/training/...   Stable_1
element * /main/LATEST -mkbranch BRANCH_NAME
```

# Miscellany

- Both branch type objects and label type objects can be locked to prevent unwanted reusal.
- 'cleartool find' is a powerful tool
- AdminVob – concept of centralized label and branch type administration
- MultiSite – concept of VOBs synchronized across the globe
  - branch masterships
  - deltas / packet synchronization

```
ct lock lbtype:Stable_0.8
```

# ClearCase

**Questions, comments?**

# Introduction to ClearCase II

v 1.2

**Vít Hrachový**

**Prague, 26 September 2007**

## Running X Applications

### Running Remote Applications

1) Start X server (if needed)

2) get your local IP address

3) log in to the remote machine - via: telnet, ssh, rsh ...

4) set the DISPLAY variable

5) run your favourite GUI application :-)

**X server** (your PC)
- start X (Xfre86e, Reflection)
- get IP (/sbin/ifconfig, ipconfig)

**X client**
set display to your IP, e.g.
- export DISPLAY=`<IP_ADDRESS>`:0

## X settings

### X Server settings

- edit ~/.Xdefaults  to suit your needs (colors, fonts)
- create script to automatically update ~/.display when you boot/login to your PC

### Reflection

- set it to run  /usr/local/bin/xterm
- and on UNIX: export COLORTERM=yes

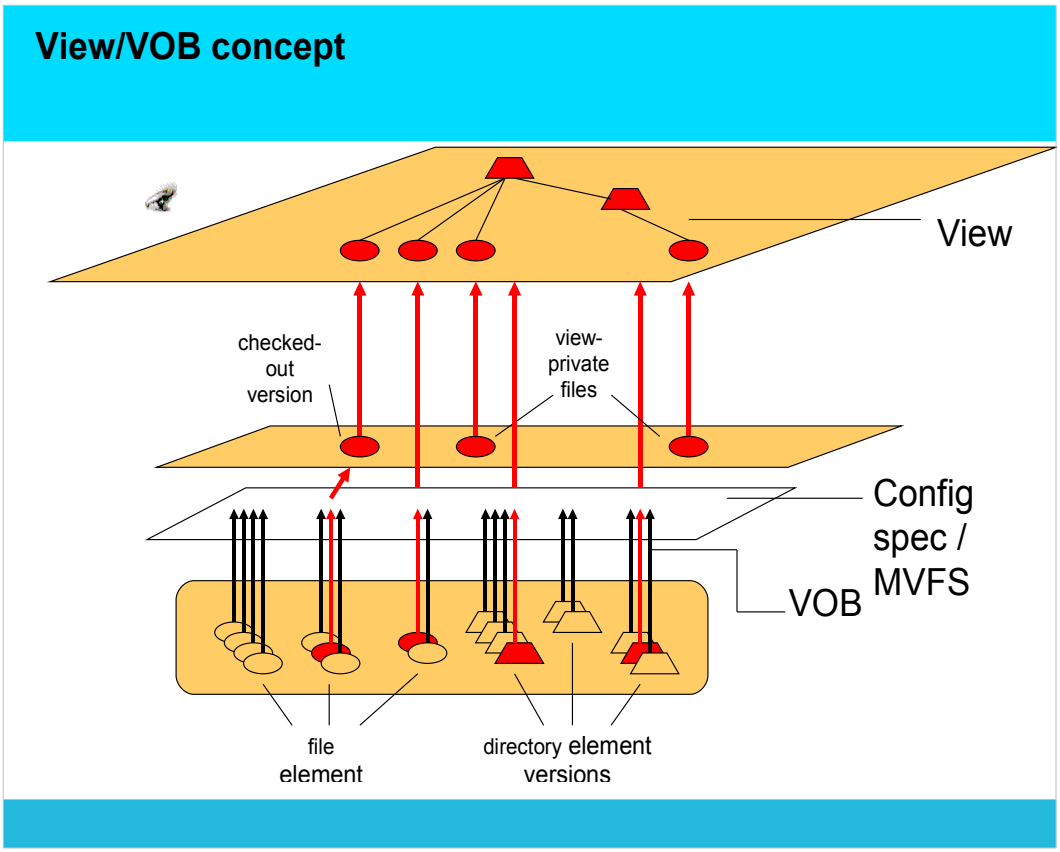
- set keys:  BackSpace  to generate ^H
  Delete  to generate ^[[3~  (or ^?)

## Working with ClearCase CLI

1. **Log in to the ClearCase host**
2. **(if not automated by your rc scripts)**
   **Set your DISPLAY, add PATH /usr/atria/bin, create the 'ct' shortcut,** e.g.:
   - bash
   - export DISPLAY=158.234.174.114:0
   - export PATH=$HOME/bin:/usr/atria/bin:$PATH
   - mkdir -p  ~/bin
   - ln -s /usr/atria/bin/cleartool ~/bin/ct
3. **(if not automated by your rc scripts)**
   **Set your view**
   - ct setview VIEW_NAME  # the same as our login name

## ClearCase - Getting Help

1. **GUI Help**
   - **xclearcase/ClearCase Explorer/ClearCase Home**
     - menu Help (requires web browser, same source as 3.)
2. **CLI Help**
   - **ct** <*command*> ***-h*** # short online help
   - **ct man** <*command / or something:-)*>
     - (e.g.: cleartool ci config_spec findmerge mkbrtype)
3. **WWW**
   - Documentation is available in form of web pages




• For a snapshot view, the user would just be seeing a local copy of each element, so just the top rectangle showing the file system as seen by the user would depict this.

## Creating new Element

1. **Ensure, that your config-spec is correct**
   - ct setcs -default  # set the default - only for Training purposes
   - ct catcs  # print current CS
2. **Check out the directory**
   - cd /data/ccase/training/`whoami`/
   - ct co -c 'adding new index.html' **.**
3. **Make an element, edit it**
   - ct mkelem -nc  index.html
   - vi index.html
4. **Check in the directory and the element**
   - ct lsco  # list the checked-out files/directories )
   - ct  ci  **.**  index.html

## Removing Element

- ct rmname – removes element references from actual directory version
  - Check out the directory
    - cd /vobs/training/`whoami`/
    - ct co -c 'CR12345 – remove obsolete files' **.**
  - Remove the files, links, directories
    - ct rmname -nc obsolete-file obsolete_link obsoleteDir
  - Check in the directory
    - ct ci -nc **.**
- ct rmelem – destroys complete element with all its history
  - DON'T use rmelem at all, unless you really know what you're doing!!!

## Information about Elements

1. **Information about the current version of a file**
   - ct describe index.html
   - ct describe -g index.html # GUI version - most commands accept -g
2. **History of an element**
   - ct lshist index.html
3. **Version tree**
   - ct lsvtree [ -g ] index.html
4. **What is the current version?**
   - ct ls
   - ct ls -d /data/ccase/training # -d = directory itself, not contents

## Test 1

1. **Change the index.html file. Add your name to it.**

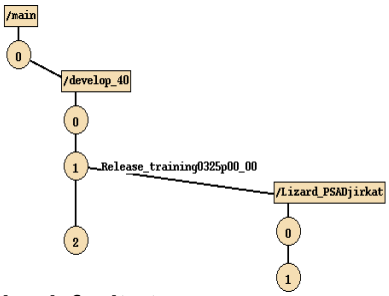
   - Tip:
     - checkout the file
     - edit it
     - check it in

```
ct co -nc index.html
vi index.html
ct ci -c 'name added' index.html

( ct describe index.html )
```

## Branches

### Branches are used for:




1. The default stream for a product / component (e.g. “HAB_0.1”); - stable branch
2. Customer specific files, different languages, etc. (e.g. “HAB_0.1_German”)
3. Task branches - to be merged to the default stream as soon as they are stable. (e.g. “XSLT_fix_HABD0123456” or “Lizard_HABD0123456”) - Remedy ticket no.
4. Bug fixes of an old version (e.g. “Rel_1.0_Fixes”, with Lizard_HABD012346 sub-branches)

## Creating Branches

1. **Create a branch type**
   - ct mkbrtype BRANCH_NAME
2. **Set your CS, so you can see the branch (latest version on it)**
3. **and if wanted - set your CS, so the branch is automatically created during a check-out**
   - ct edcs # edit config spec

```
element * CHECKEDOUT
element * .../BRANCH_NAME/LATEST
element * /main/LATEST -mkbranch BRANCH_NAME
```

4. **The branch on a specific file will be created when you**
   - ct mkbranch -nc BRANCH_NAME index.html # or
   - ct co index.html # or
   - ct mkelem new_file

## Test 2

1. **Create a German version of index.html on a branch "HAB_0.1_German" and**
2. **Create a new file "default.css" (on the same branch)**

```
ct mkbrtype HAB_0.1_German
ct edcs     # see below
ct co -nc index.html
vi index.html
ct ci -nc index.html

ct co .
ct mkelem default.css
vi default.css
ct ci -nc . default.css
```

```
element * CHECKEDOUT
element * .../HAB_0.1_German/LATEST
element * /main/LATEST -mkbranch HAB_0.1_German
```

## Test 3

1. **Set config. spec to show the latest version of the main branch.**
2. **List the directory contents**
3. **Where did the 'default.css' file go ?**
   - Tip:
     - save your current config spec to a file:
       ct catcs **>** ~/branch_config_spec.cs
       (you can reuse it later by: ct setcs FILE)

```
ct edcs     # edit CS
ls
#or: ct setcs -default
#the file is in a different
version of the current
directory
```

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

## Merge

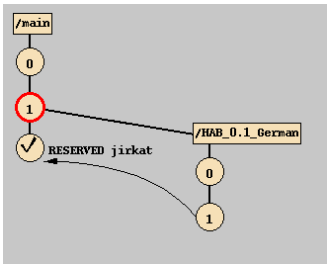
- **Merging changes**
  - set config spec to the **target** version
    **ct edcs**

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

  - create a batch file
    **ct findmerge . -fver .../Lizard_PSADjirkat/LATEST -print**
  - or run graphical merge
    **ct findmerge . -fver .../Lizard_PSADjirkat/LATEST -gmerge**
  - or run merge manually
    **ct co best.c**
    **ct merge -g -to index.html **
    **index.html@@/main/HAB_0.1_German/LATEST**

## Merge

- **Merge result**
  - check in
    **ct ci index.html**



## Labels

**Label is a symbolic name of a certain version of an element**

- There must be a label type defined, before the label can be used.

# make a label type

- ct mklbtype Stable_1

# place it on the current versions of the elements

- ct mklabel Stable_1 default.css index.html .

# look at the version tree

- ct lsvtree -g index.html

## Labels

**In config. spec, you can refer to a version using a label**

```
element * CHECKEDOUT
# for all elements under /vobs/cm/training
# show the version labeled "Stable_1"
element /vobs/cm/training/...   Stable_1
element * /main/LATEST -mkbranch BRANCH_NAME
```

## Miscellany

- Both branch type objects and label type objects can be locked to prevent unwanted reusal.
- 'cleartool find' is a powerful tool
- AdminVob – concept of centralized label and branch type administration
- MultiSite – concept of VOBs synchronized across the globe
  - branch masterships
  - deltas / packet synchronization

```
ct lock lbtype:Stable_0.8
```

## ClearCase

**Questions, comments?**