File: vendor5readme.pdf
Purpose: 1. To present an overview on how to merge or track vendor (third-party) software when developing local code. The overview includes an outline of a detailed example used to show the vendor merge capablitiy of tkcvs
2. To define a set of code to be used for the example.
3. To describe script vendorcode.sh which sets up the source code used for the example.
4. To describe the use of a version of tkcvs subsequent to 7.2 for importing the code created in vendorcode.sh.
5. To describe the use a version of tkcvs subsequent to 7.2 for merging in vendor or third-party source into the local code.

By: Eugene Lee, Aerospace Corporation, 10/4/95
Revised: EAL 7/6/01 for use with TkCVS 7.0
Revised: EAL 7/11/01 for use with tkcvs_gene
Revised: EAL 1/23/04 for use with a version of tkcvs subsequent to 7.2 (1/3/04)

1. Software development is sometimes based on source distribution from a vendor or third-party distributor. After building a local version of this distribution, merging or tracking the vendor's future release into the local version of the distribution can be done with tkcvs. The following example will demonstrate the vendor merge capability of tkcvs.
   a. Vendor's Release 1.0 is imported into CVS as Module 3rdParty with version number 1.0.
   b. A copy of the Vendor's Release 1.0 was modified with local changes. This local version is imported into CVS as Module Local-1.0 with version number 1.0.
   c. Vendor's Release 1.1 is imported into the existing Module 3rdParty with version number 1.1. (Importing to an existing module is a new operation subsequent to tkcvs 7.2).
   d. From a new directory, Module Local-1.0 is checked out. The differences between versions 1.0 to 1.1 of Module 3rdParty are merged into the local code. Local changes are made which will later cause a conflict with the next release of the Vendor's code. This version of the local code is imported into CVS as Module Local-1.1 with version 1.1.
   e. Vendor's Release 2.0 is imported into the existing Module 3rdParty with version number 2.0.
   f. From a new directory, Module Local-1.1 is checked out. The differences between versions 1.1 to 2.0 of Module 3rdParty are merge into the local code. The resulting updated local code will include a merge conflict which the user must resolve. After that is done this version of the local code is imported into CVS as Module Local-2.0 with version 2.0

   Note: Each of the updated local code was imported into the CVS repository as a separate new module, i.e., Local-1.1 and Local-2.0. This was done for simplicity. To conserve resources, all the local code could have been imported into the same module, i.e., Module Local using different version numbers. The example would have been a little more complex to describe.

2. To test out the merging or tracking of vendor (third-party) source releases using CVS, the following set of pseudo code is presented. This pseudo code was used so that a listing of all the routines can be easily shown in this document.

```
############################## 3rdParty Code ##############################
#              |     main     |    get       |    sort      |   getsort    #
#______________|______________|______________|______________|______________#
# Release 1.0  | program Main | proc Get     |              |              #
#              | Release 1.0  | Release 1.0  | (undefined)  | (undefined)  #
#              |  .           |  ..          |              |              #
#              |  ..          |  ..          |              |              #
#              | Get          | end          |              |              #
#              |  ..          |              |              |              #
#              | end          |              |              |              #
#______________|______________|______________|______________|______________#
# Release 1.1  | program Main | proc Get     | proc Sort    |              #
#              | Release 1.1  | Release 1.1  | Release 1.1  | (undefined)  #
#              |  .           |  ..          |  ..          |              #
#              |  ..          |  ..          | end          |              #
#              | Get          |  (new code)  |              |              #
#              |  ..          | end          |              |              #
#              | Sort         |              |              |              #
#              | Printout     |              |              |              #
#              | end          |              |              |              #
#______________|______________|______________|______________|______________#
# Release 2.0  | program Main |              |              | proc GetSort #
#              | Release 2.0  |  (deleted)   |  (deleted)   | Release 2.0  #
#              |  .           |              |              |  ..          #
#              |  ..          |              |              |  ..          #
#              | GetSort      |              |              | end          #
#              | Printout     |              |              |              #
#              | end          |              |              |              #
############################## 3rdParty Code ##############################
```

3rdParty Code: In Release 1.0 only program Main and proc Get are defined.

In Release 1.1 program Main and proc Get are modified and proc Sort is added.

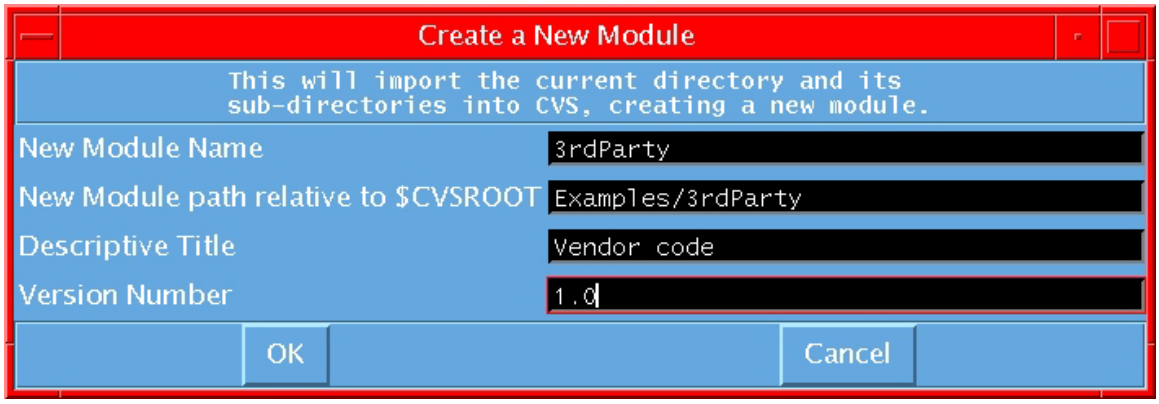
In Release 2.0 proc GetSort is added to replace the functions previously provided by procs Get and Sort. Procs Get and Sort are deleted. Program Main is modified.

(Note: All 3rdParty routines are commented with the current release numbers.)

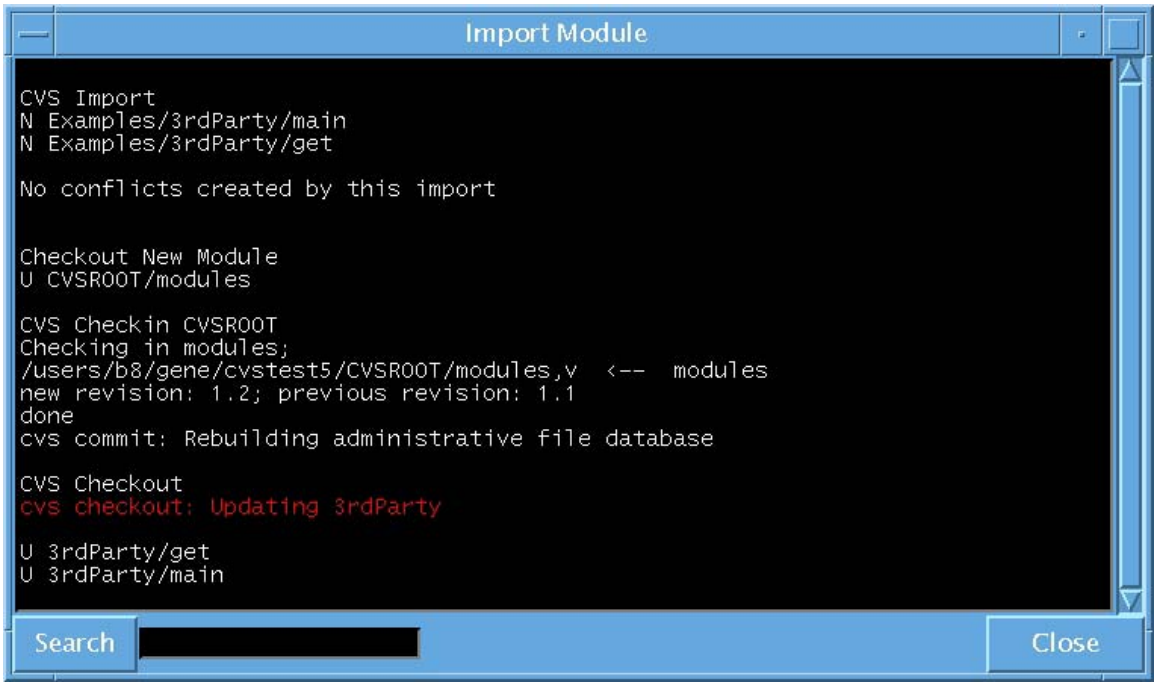
The changes between releases 1.0, 1.1, and 2.0 includes new and deleted routines. The purpose of introducing new and deleted routines is to show how to handle them when using tkcvs. Also there is a change to main in the local code between Release 1.1 and 2.0. which will create a merge conflict in file main when the local code is updated to 3rdParty release 2.0. This will be used to show the format CVS uses for delineating code which are in conflict after a merge.

```
########### Local Code Release 1.0 ###########
#            |     main        |    get       #
#____________|_________________|______________#
# Release 1  | program Main    | proc Get     #
#            | Release 1.0     | Release 1.0  #
#            |  .              |  ..          #
#            |  . (my code)    |  ..          #
#            |  ..             |              #
#            | Get             | end          #
#            |  ..             |              #
#            | end             |              #
########### Local Code Release 1.0 ###########
```

```
Local Code: In Release 1.0, the local code is identical to the 3rdParty
            Code, Release 1.0, except for the ". (my code)" before the Get
            statement.

-------------------------------------------------------------------------------
3. Script vendorcode.sh will create the source code to be used for the example.

To use this script:

  o From a working directory, i.e., ~/tmp_vgen :

    o Copy file vendorcode.sh into the working directory and invoke
      vendorcode.sh.
      o This will create the source code in the following directories
        ~/tmp_vgen/Examples/Local-1.0   <- local code release 1.0
        ~/tmp_vgen/Examples/3rdPartyV1  <- vendor code release 1.0
        ~/tmp_vgen/Examples/3rdPartyV2  <- vendor code release 1.1
        ~/tmp_vgen/Examples/3rdPartyV3  <- vendor code release 2.0

-------------------------------------------------------------------------------
4. Use tkcvs to import the example code into CVS modules

  o If you have never used CVS and wish to create a new test CVS
    for experimenting, do the following:

    o From your home directory
      unsetenv CVSROOT
      cvs -d ~/cvstest init  <- creates & initializes CVS repository
      setenv CVSROOT ~/cvstest


    o Use tkcvs to import release 1.0 of the 3rdParty code into a new module
      named 3rdParty.
      o cd to ~/tmp_vgen/Examples/3rdPartyV1
      o invoke tkcvs_gene4
        o From the TkCVS file menu, select "Browse Modules"
        o From the Module Browser file menu, select "Import To A New Module"
        o In the "Create a New Module" window, enter the following:
          New Module Name                      : 3rdParty
          New Module path relative to $CVSROOT : Examples/3rdParty
          Descriptive Title                    : Vendor code
          Version Number                       : 1.0  <- same as release no.
```



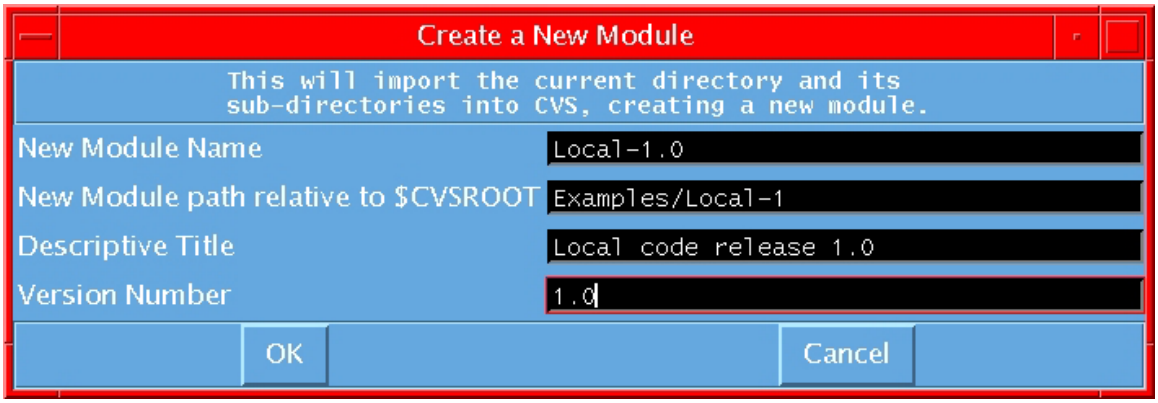

```
        o Press the OK button.
        o The following window will appear which shows that the code in
          directory 3rdPartyV1 was imported into the CVS repository. tkcvs will
          then move the original 3rdPartyV1 directory to 3rdPartyV1.orig. Next,
          tkcvs will check out the new module into directory 3rdParty
```



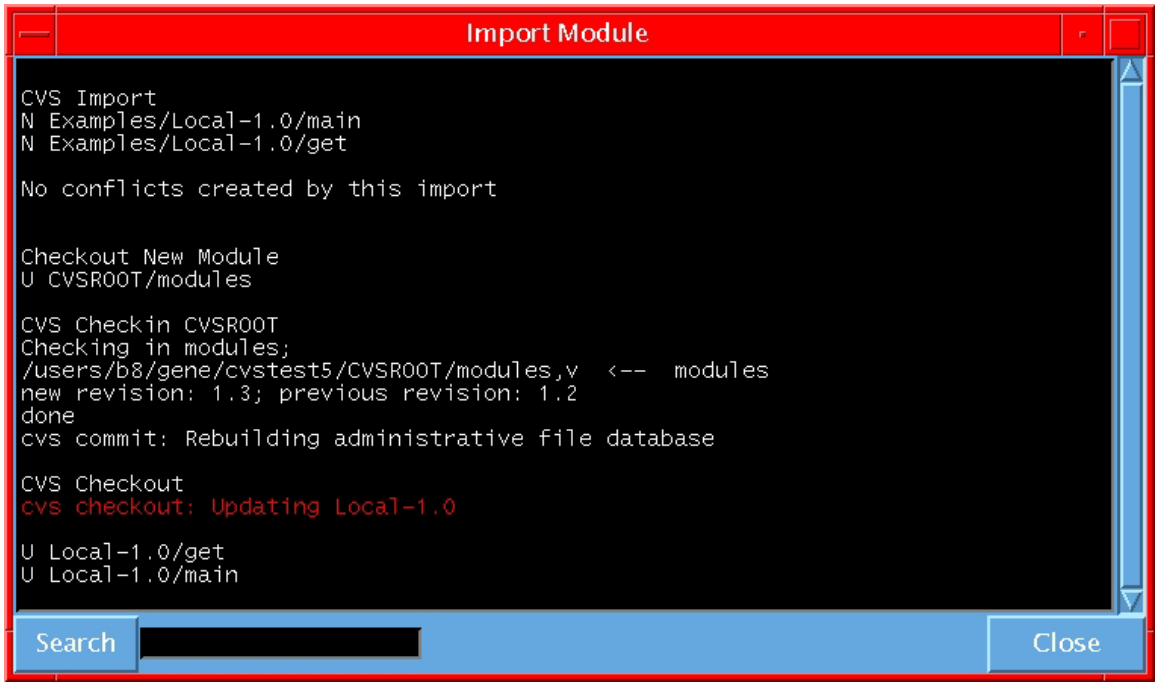
- Press the Close button in the Import Module window.
- From the Module Browser, click on the + Examples directory icon to show that module 3rdParty is now in the repository.
- Exit tkcvs.
- Note that you are now in directory 3rdPartyV1.orig.

- Use tkcvs to import in the local code (which has one statement in file main which is not in the vendor's original code) into a new module named Local-1.0.
  - cd to ~/tmp_vgen/Examples/Local-1.0
  - invoke tkcvs
    - From the TkCVS file menu, select "Browse Modules"
    - From the Module Browser file menu, select "Import To A New Module"
    - In the "Create a New Module" window, enter the following:

      New Module Name : Local-1.0
      New Module path relative to $CVSROOT : Examples/Local-1.0
      Descriptive Title : Local code release 1.0
      Version Number : 1.0 <- same as release no.

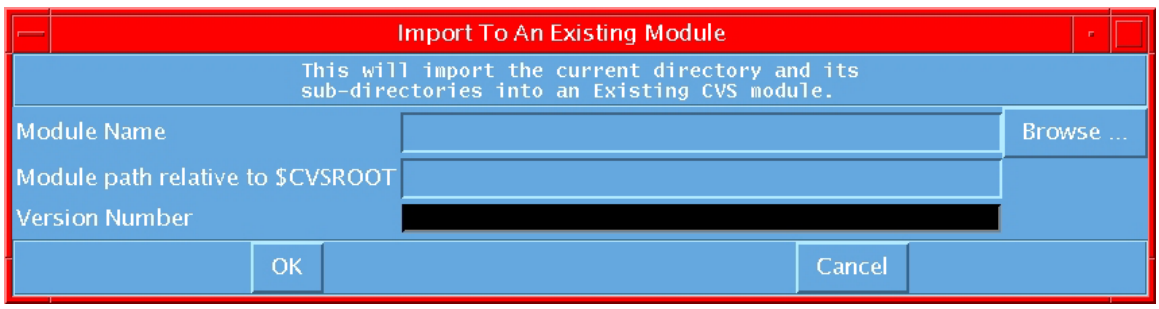


- Press the OK button.
- The following window will appear which shows that the code in directory Local-1.0 was imported into the CVS repository. tkcvs will then move the original Local-1.0 directory to Local-1.0.orig. Next, tkcvs will check out the new module into directory Local-1.0 (that is why the original Local-1.0 directory was moved to directory Local-1.0.orig).

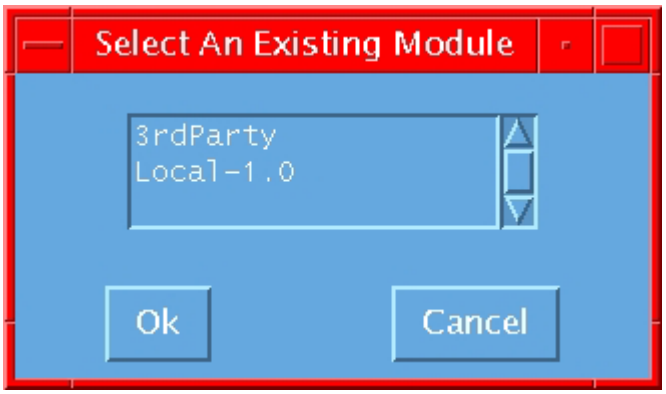


- Press the Close button in the Import Module window.
- From the Module Browser, click on the + Examples directory icon to show that module Local_1.0 is now in the repository.
- Exit tkcvs.
- Note that you are now in directory Local-1.0.orig.

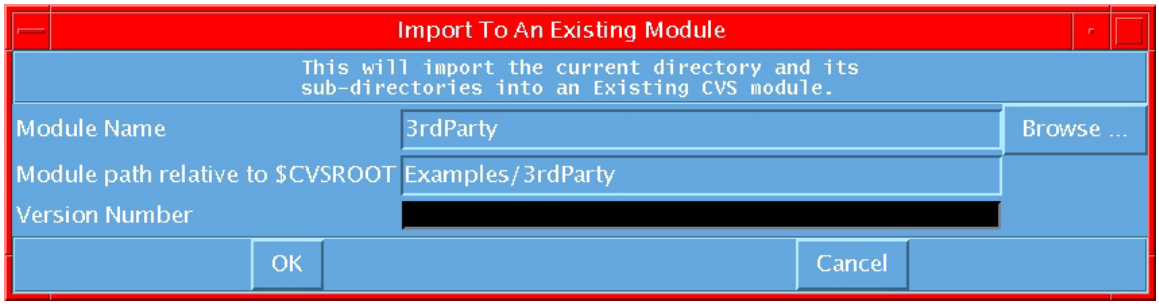
- Use tkcvs to import release 1.1 of the vendor code into existing Module 3rdParty.
  - cd to ~tmp_vgen/Examples and remove directory 3rdParty, i.e., rm -rf 3rdParty
  - cd to ~/tmp_vgen/Examples/3rdPartyV2
  - invoke tkcvs
    - From the TkCVS file menu, select "Browse Modules"
    - From the Module Browser press the + Examples directory icon so that Modules Local-1.0 and 3rdParty appears.
    - From the Module Browser file menu, select "Import To An Existing Module"



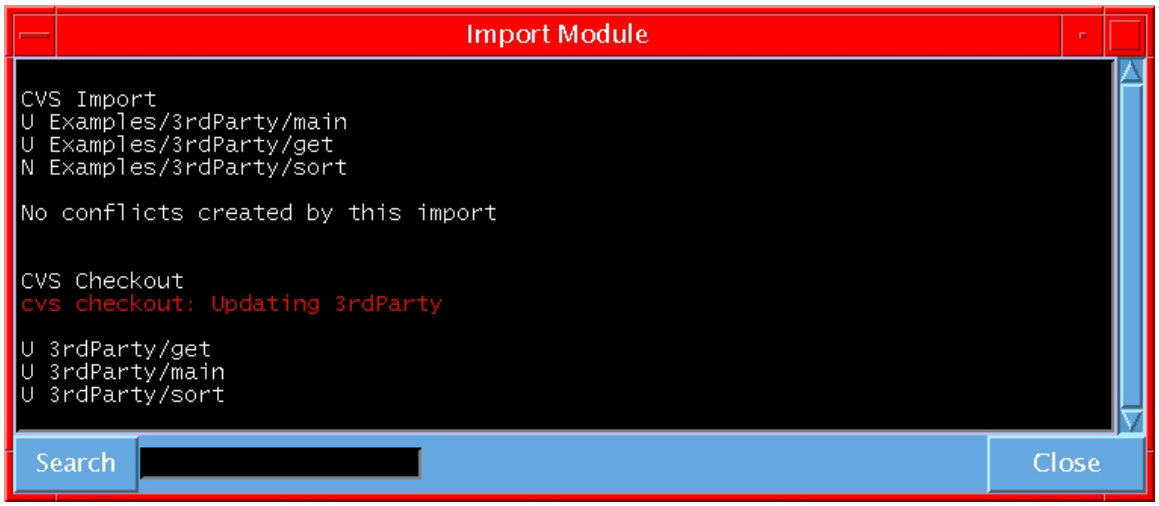
- Press the "Browse" button which will prompt you to select an existing module



- Select the 3rdParty module and press the OK button.

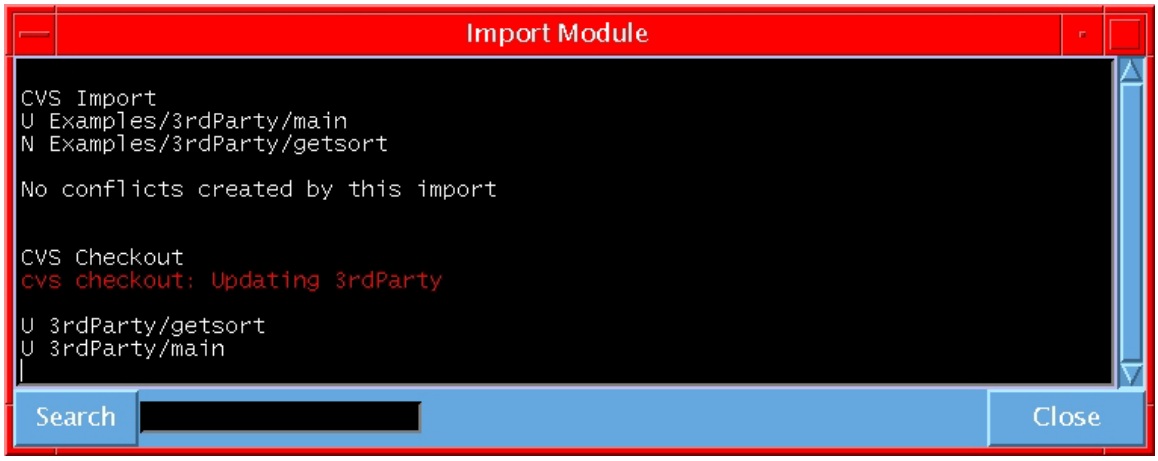


- From the "Import To A Existing Module" window input "1.1" for the Version Number and then press the OK button.
- The following window will appear which shows that the code in directory 3rdPartyV2 was imported into the CVS repository. tkcvs will then move the original 3rdPartyV2 directory to 3rdPartyV2.orig. Next, tkcvs will check out version 1.1 of Module 3rdParty.



- Press the Close button to close the Import Module window.
- Exit tkcvs.
- Note that you are now in directory 3rdPartyV2.orig.

- Use tkcvs to import release 2.0 of the vendor code into existing Module 3rdParty.
  - cd to ~tmp_vgen/Examples and remove directory 3rdParty, i.e., rm -rf 3rdParty
  - cd to ~/tmp_vgen/Examples/3rdPartyV3
  - invoke tkcvs
    - From the Module Browser press the + Examples directory icon so that Modules Local-1.0 and 3rdParty appears.
    - From the Module Browser file menu, select "Import To An Existing Module"
    - From the "Import To An Existing Module" window press the Browse button, select module 3rdParty and then press the OK Button.
    - From "Import To An Existing Module" input 2.0 for the version Number and then press the OK Button
    - The following window will appear which shows that the code in directory 3rdPartyV3 was imported into the CVS repository. tkcvs will then move the original 3rdPartyV3 directory to 3rdPartyV3.orig. Next, tkcvs will check out version 2.0 of Module 3rdParty.



    - Press the Close button to close the Import Module window.
    - Exit tkcvs.
    - Note that you are now in directory 3rdPartyV3.orig.

---

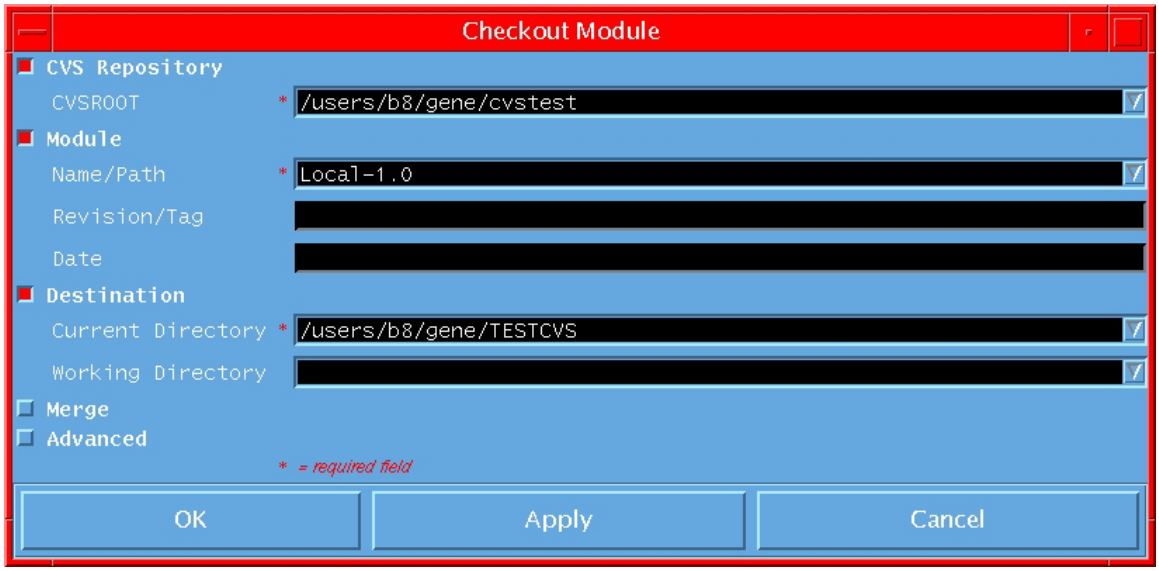
5. Use tkcvs to merge in code from the vendor or third-party into the local code.

a. Merge in code from revision 1.0 to 1.1 of Module 3rdParty into local code from Module-1.0. From an empty working test directory, i.e., ~/TESTCVS, (not ~/cvstest), type:

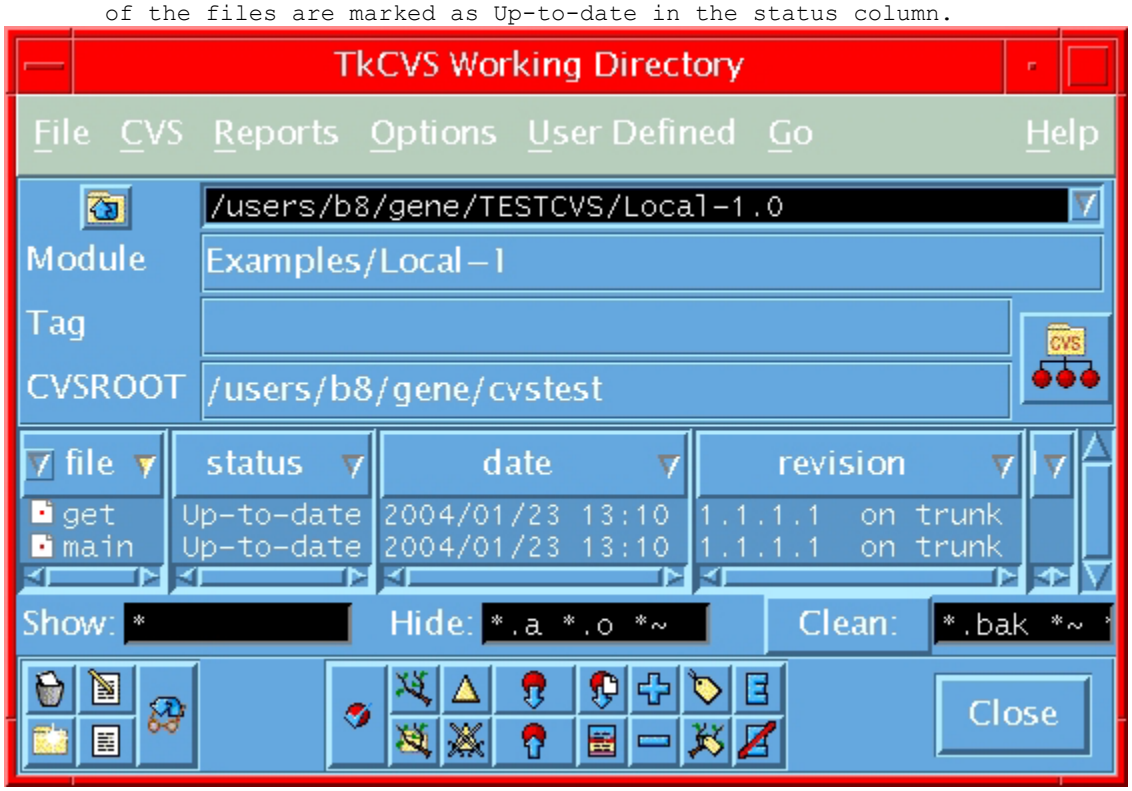
```
setenv CVSROOT ~/cvstest
tkcvs
```

First the code from module Local-1.0 will be checked out.

- From the TkCVS file menu, select "Browse Modules"
- From the Module Browser, click on the Examples + icon which will then show modules Local-1.0 and 3rdParty.
- Press module Local-1.0. The Module entry box should show "Local-1.0".
- Press the "Check out a module from the repository" button located 4th from the left at the bottom of the window.



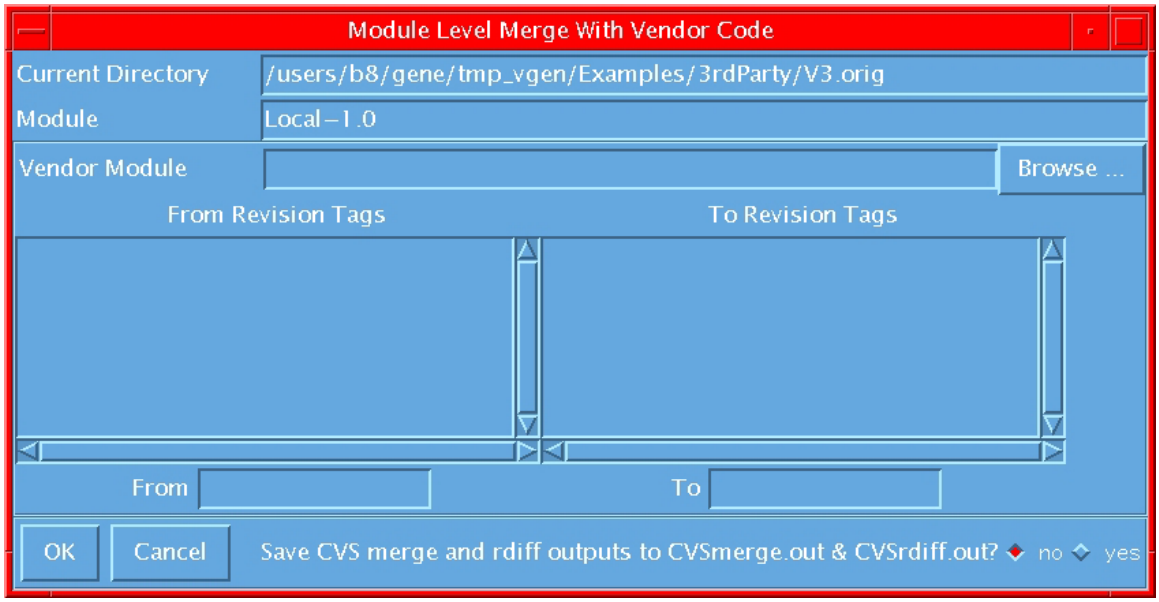
- Press the OK button. Window "Confirm!" will ask "This will check out Local-1.0 from CVS. Are you sure?" Press the OK button.
- Window "CVS Checkout" will be opened. This is the output from CVS when checking out a module. After reviewing its content, press the Close button.
- In the TkCVS window refresh the window by pressing on the "Re-read the current directory" button (the one with a picture of an eyeglass). Directory Local-1.0 will appear with a CVS icon.
- Double click on the CVS icon which will bring you into directory Local-1.0 in your working test directory TESTCVS. Note that each of the files are marked as Up-to-date in the status column.



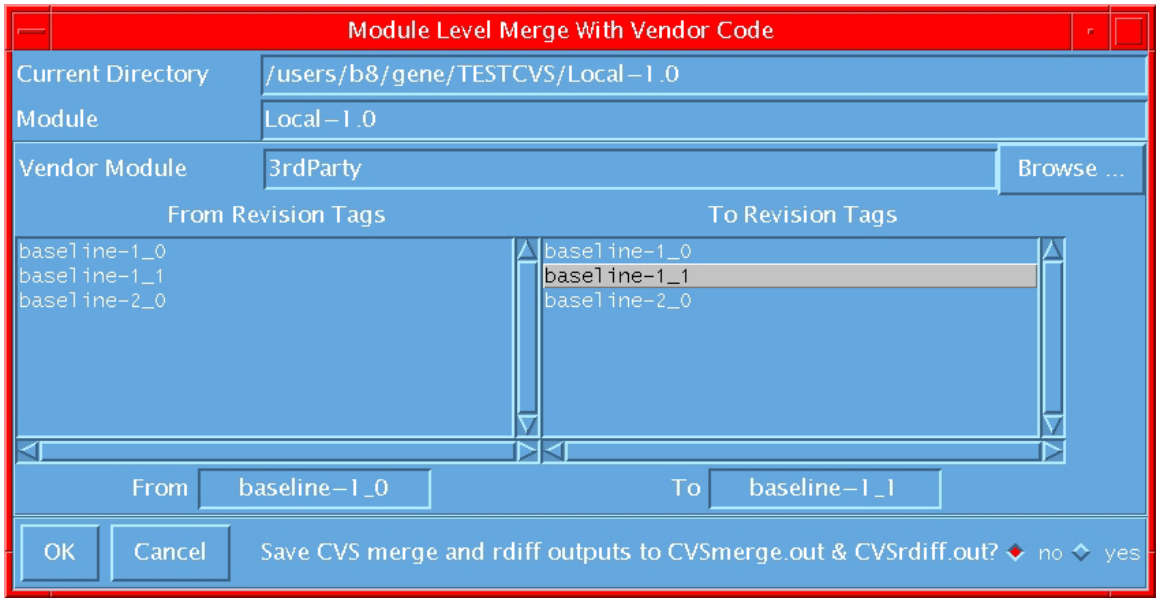
- Select file main and press the "Edit the selected files" button (icon with pencil & paper) to verify that the code matches that shown Section 2. Do that for file get also.

Merge in differences between Release 1.0 and 1.1 of the 3rdParty code into the working copy of the Local-1.0 code.

- Open up the Module Browser and press the + icon on the Examples directory
- Press module Local-1.0. The Module entry box should show "Local-1.0".
- From the Module Browser file menu, select "Vendor Merge"



- Press the "Browse" button.
- From the "Select A Vendor" window, select the 3rdParty module and press the OK button.
- The "Module Level Merge With Vendor Code" window will now show (which may take several seconds) the revision tags in the two scroll lists.



Note: Each time a tkcvs import command is invoked, the set of files imported is tagged with a tag of the form baseline-xx, where

xx is of the version number with each "." replaced with "_",
i.e., 1.0 becomes 1_0

- From the left scroll list click on "baseline-1_0" so that the selection will be placed into the "From" entry box
- From the right scroll list click on the "baseline-1_1" so that the selection will be placed into the "To" entry box.
- There is an option for saving the output from the CVS merge and rdiff. For this exercise leave this selection at "no".
- Press the OK button.
- You will be prompted to verify that you want to merge differences between baseline-1_0 and baseline-1_1 of 3rdParty into Local-1.0 Press the OK button

- Multiple windows will then appear showing outputs from intermediate steps.

  - One will be the CVS output from checking out temp files used by tkcvs in preparation for doing the merge. Press the Close button.

  - One will be the CVS output from performing the merge. The title of this window shows that module 3rdParty has been merged into the checked out copy of module Local-1.0. The contents of this window when I ran my copy of the example is:

```
RCS file: /users/b8/gene/cvstest/Examples/3rdParty/get,v
retrieving revision 1.1.1.1
retrieving revision 1.1.1.2
Merging differences between 1.1.1.1 and 1.1.1.2 into get
M /users/b8/gene/TESTCVS/Local-1.0/main
RCS file: /users/b8/gene/cvstest/Examples/3rdParty/main,v
retrieving revision 1.1.1.1
retrieving revision 1.1.1.2
Merging differences between 1.1.1.1 and 1.1.1.2 into main
U /users/b8/gene/TESTCVS/Local-1.0/sort
cvs checkout: Updating /users/b8/gene/TESTCVS/Local-1.0
```

As you can see, files get and main were merged. File sort was introduced by the 3rdParty at Release 1.1. It was checked out to this working directory as shown by the U in column 1. Press the Close button.

- One will be the CVS output from doing a rdiff. The contents of this window is:

```
File Examples/3rdParty/get changed from revision 1.1.1.1 to 1.1.1.2
File Examples/3rdParty/main changed from revision 1.1.1.1 to 1.1.1.2
File Examples/3rdParty/sort is new; current revision 1.1.1.1
cvs rdiff: Diffing Examples/3rdParty
```

Press the Close button.

- In the TkCVS window, after pressing the "Re-read the current directory" button, the status of files get and main are now changed to Locally Modified and status of file sort is ?.
- Select file main and press the "Edit the selected files" button to verify that the changes between Release 1.0 and 1.1 of the 3rdParty code has been correctly merged into the working directory. Do that for file get also.
- Since file sort is new to the working directory its status is shown by ? mark.
- This completes the merge of the code between Release 1.0 and 1.1 of the 3rdParty code into the checked out copy of the local code

from module Local-1.0.

- Make a local change in file main so that a merge conflict will occur later when doing a vendor merge to a later version of the 3rdParty code.
- Either from the edit mode with tkcvs, or outside of tkcvs, edit the line in file main so that the line

```
Sort
```

is changed to

```
Sort1(my code)
```

As stated earlier, this updated local code will not be check back into an existing local module in the repository. Below, it will be imported into a new module named Local-1.1. Exit tkcvs.

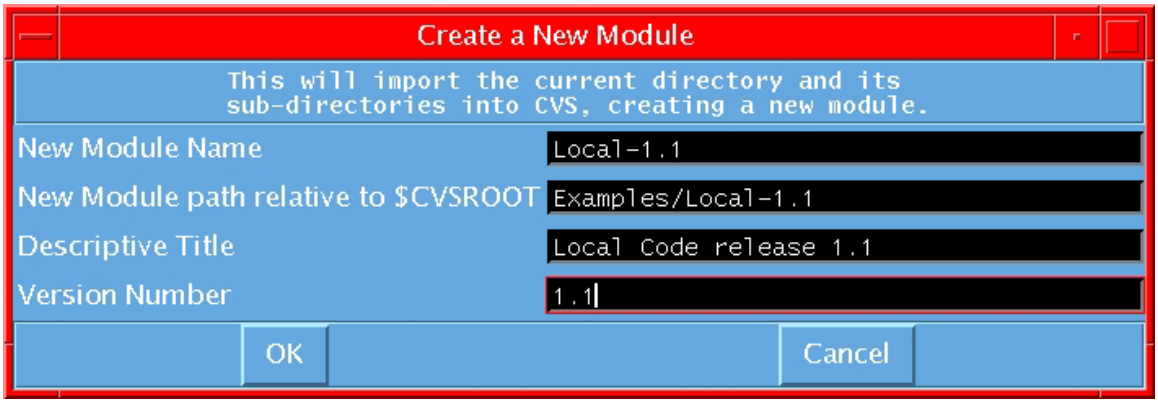
Import the local changes to module Local-1.1

- From the working test directory where the above changes were made there should only be a directory named Local-1.0. cd to directory Local-1.0 and invoke the ls command. You should see:

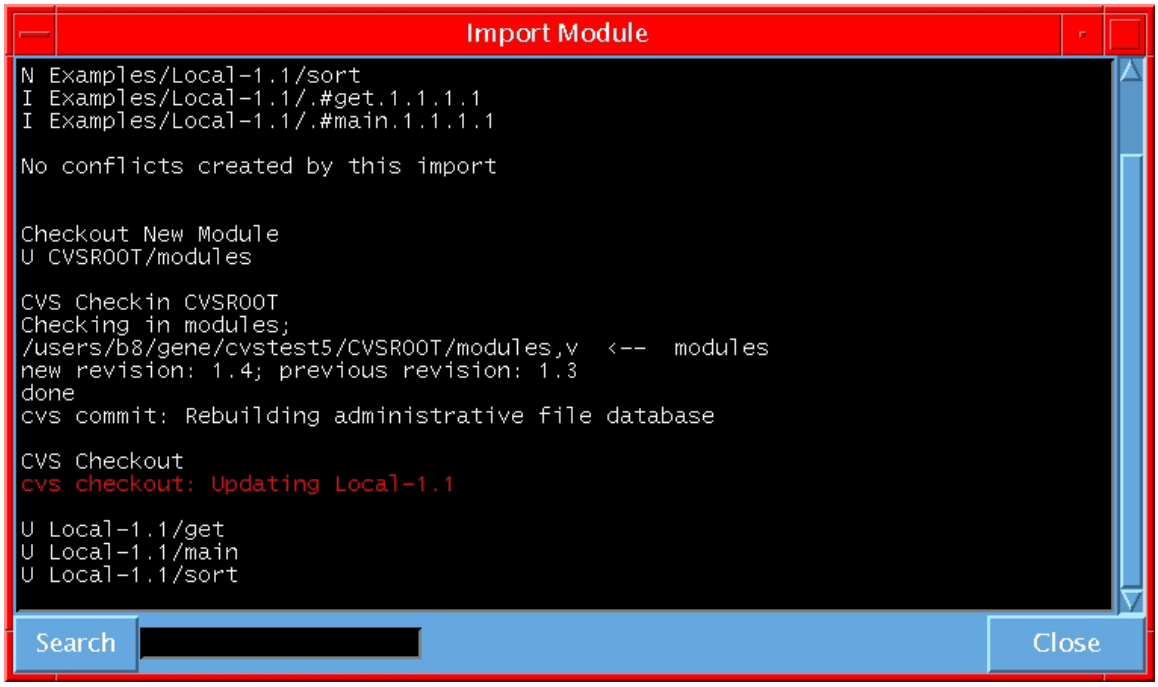
```
CVS/    get     main    sort
```

- Remove the CVS directory with the command, rm -rf CVS. The files in this directory are no longer under CVS. It is a working directory with the original Local-1.0 code modified by the merge with 3rdParty code plus a one line change in file main.
- Invoke tkcvs. The file scroll list will show files get, main, and sort.
- From the TkCVS file menu, select "Browse Modules"
- In the module browser filemenu select “Import To A New Module”.
- In the "Create a New Module" window enter the following:

```
New Module Name                       : Local-1.1
New Module path relative to $CVSROOT  : Examples/Local-1.1
Descriptive Title                     : Local code release 1.1
Version Number                        : 1.1
```



- Press the OK button.
- The following window will appear which shows that the code in the working directory was imported into the CVS repository. tkcvs will then move directory ~/CVSTEST/Local-1.0 to ~/CVSTEST/Local-1.0.orig. Next, tkcvs will check out the new module into directory Local-1.1.



- Press the Close button.
- From the Module Browser, click on the + Examples directory icon to verify that module Local_1.1 is now in the repository.
- Exit tkcvs.
- Note that you are now in directory Local-1.0.orig. Remember that code in ~/CVSTEST/Local-1.0.orig is original Local-1.0 code modified by the merge with 3rdParty code plus a single line change in file main.)

b. Merge in code from revision 1.1 to 2.0 of Module 3rdParty into a working copy of the Local-1.1 code. Since module Local-1.1 was checked out above in directory ~/TESTCVS/Local-1.1, cd to that directory and type tkcvs.

- Open up the Module Browser and press the + icon in the Examples directory.
- Press module Local-1.1. The Module entry box should show "Local-1.1".
- From the Module Browser file menu, select "Vendor Merge"
- The "Module Level Merge With Vendor Code" window will appear.
- Press the "Browse" button.
- In the "Select A Vendor" window select module 3rdParty and press the OK button.
- The "Module Level Merge With Vendor Code" window will now show the revision tags in the two scroll lists.
- From the left scroll list click on "baseline-1_1" so that the selection will be placed into the "From" entry box.
- From the right scroll list click on the "baseline-2_0" so that the selection will be placed into the "To" entry box.
- Press the OK button.
- You will be prompted to verify this is what you want. Press OK.
- Multiple windows will then appear showing outputs from intermediate step.
  - One will be the CVS output from checking out temp files used by tkcvs in preparation for doing the merge. Press the Close button.

o One will be the CVS output from performing the merge. The title of this window shows that module 3rdParty has been merged into the checked out copy of module Local-1.1. The contents of this window when I ran my copy of the example was:

```
--------- This is what appears but some of the lines are out of order --------
U /users/b8/gene/TESTCVS/Local-1.1/getsort                                    -
M /users/b8/gene/TESTCVS/Local-1.1/main                                       -
RCS file: /users/b8/gene/cvstest/Examples/3rdParty/main,v                     -
retrieving revision 1.1.1.2                                                   -
retrieving revision 1.1.1.3                                                   -
Merging differences between 1.1.1.2 and 1.1.1.3 into main                     -
cvs checkout: Updating /users/b8/gene/TESTCVS/Local-1.1                       -
cvs checkout: file /users/b8/gene/TESTCVS/Local-1.1/get is locally modified, -
              but has been removed in revision baseline-2_0                   -
rcsmerge: warning: conflicts during merge                                     -
cvs checkout: file /users/b8/gene/TESTCVS/Local-1.1/sort is locally modified,-
              but has been removed in revision baseline-2_0                   -
-------------------------------------------------------------------------------

----- This is what should have appear (when I used CVS from the command ------
      line outside of tkcvs)
cvs checkout: Updating /users/b8/gene/TESTCVS/Local-1.1
cvs checkout: file /users/b8/gene/TESTCVS/Local-1.1/get is locally modified,
              but has been removed in revision baseline-2_0
U /users/b8/gene/TESTCVS/Local-1.1/getsort
M /users/b8/gene/TESTCVS/Local-1.1/main
RCS file: /users/b8/gene/cvstest/Examples/3rdParty/main,v
retrieving revision 1.1.1.2
retrieving revision 1.1.1.3
Merging differences between 1.1.1.2 and 1.1.1.3 into main
rcsmerge: warning: conflicts during merge
cvs checkout: file /users/b8/gene/TESTCVS/Local-1.1/sort is locally modified,
              but has been removed in revision baseline-2_0

(The following explanation will use the output that should have appeared.
 I'll try to find out why the output is out of order later.)
```

Files get and sort has been removed in baseline-2_0 (Release 2.0). Make notice of this since there will not be any indication of this when you return to the TkCVS window. The output from rdiff shown next will explicitly show which files have been removed. The U associated with file getsort shows that this file was checked out and is new to the checked out copy of Local-1.1. File main has been merged but there is a warning message that it has conflicts. Press the Close button.

o One will be the CVS output from doing a rdiff. The contents of this window is:

```
File Examples/3rdParty/get is removed; not included in release tag baseline-2_0
File Examples/3rdParty/getsort is new; current revision 1.1.1.1
File Examples/3rdParty/main changed from revision 1.1.1.2 to 1.1.1.3
File Examples/3rdParty/sort is removed; not included in release tag baseline-
2_0
cvs rdiff: Diffing Examples/3rdParty
```

Press the Close button.

o In the TkCVS window, update the window by pressing the “Re-read the current directory” button. The status of file get and sort are marked as Up-to-date, even though we know that it was removed in the 3rdParty code at Release 2.0. It is up to the user to decide if these two

files should also be removed from the working test directory. The file getsort is marked by the ? mark since it is new to the working directory. File main, as noted in a previous window, has a merge conflict.

- View the file main by double clicking on "main" or use the "Edit the selected files" button. The editor will show:

```
program Main
Release 2.0
 .
 . (my code)
 ..
<<<<<<< main
Get
 ...
Sort1(my code)
=======
GetSort
>>>>>>> 1.1.1.3
Printout
end
```

What this says is that CVS could not handle the overlaps. The overlap is delineated by the lines

```
<<<<<<< main

=======

>>>>>>> 1.1.1.3
```

This file should be edited to yield:

```
program Main
Release 2.0
 .
 . (my code)
 ..
GetSort
Printout
end
```

- This completes the merge of the code between Release 1.1 and 2.0 of Module 3rdParty into the checked out copy of the working code from Module Local-1.1. As stated earlier, this updated local code will be not be checked back into the existing Module Local-1.1. Below, it will be imported into a new module named Local-2.0. Exit tkcvs.

Import the changes to module Local-2.0

- From the working test directory where the above changes were made there should only be a directory named Local-1.1. cd to Local-1.1 and invoke the ls command. You should see:

```
CVS/   get    getsort    main   sort
```

- Remove the CVS directory with the command, rm -rf CVS. The files in this directory are no longer under CVS.
- Since get and sort is no longer in the 3rdParty Release 2.0 code and will also not be used in the local code Release 2.0, removed them with the command, rm get sort
- Invoke tkcvs. The file scroll list will show files getsort and main.
- From the TkCVS file menu, select "Browse Modules"

- From the Module Browser file menu, select "Import To A New Module"
- In the "Create a New Module" window enter the following:

```
New Module Name                      : Local-2.0
New Module path relative to $CVSROOT : Examples/Local-2.0
Descriptive Title                    : Local change release 2.0
Version Number                       : 2.0
```

- Press the OK button. When the "Import Module" window appears summarizing the output of the import command, press the Close button.
- From the Module Browser, click on the + Examples directory icon to verify that module Local-2.0 is now in the repository.
- Exit tkcvs.

---

5. Comments

- This example is based on a proposed update to tkcvs version 7.2 dated 1/4/04.

- The proposed changes involves the following changes to the tkcvs source code version 7.2.

  a. Modifications to routine modbrowse.tcl to add selection for "Import To An Existing Module" and changing selection "Import" to "Import To A New Module".
  b. A new routine import2.tcl for implementing the function "Import To An Existing Module. The new code includes a robust Browse window so that the user cannot input inconsistent data.
  c. Modifications to routine import.tcl to allow the user flexibility in defining the name of the module when importing to a new module. (In Version 7.2 of tkcvs, the default name was the directory name of the source code to be imported.)

     The method for testing for a duplicate key was changed.
     Modified what is written for the #D line when making changes to the CVS module file.

     Since the name of the new module does not necessarily have to be identical to the name of the directory, the directory to be changed to after the new module is checked out will not necessarily be $cwd. Variable chmoddir was defined as the name of the directory to go to after checking out the new module.
  d. Modifications to routine merge.tcl now enables the user to select the name of the module where merge code is to be from. A robust Browse window was added so that the user cannot input inconsistent data. Proc vendor_wait was replaced with proc vendorDialog.

     An option was added to so that the output from the CVS merge and rdiff can be saved to output files. They will be useful when there are many files in a module. The user can peruse these output files to double check for new, deleted, and conflicting files.

  e. Modification to routine help.tcl to add help on the "Import To An Existing Module" and "Vendor Merge" operations.
  f. Modification to routine workdir.tcl to define calls to help for the "Import To An Existing Module" and "Vendor Merge" operations.