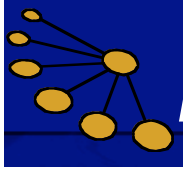
CVS - Concurrent Versions System Crash-Course

Gerhard Klimeck, Michael McLennan
Purdue University

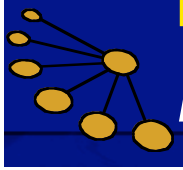
- Who should care?
- What is a code revision system?
- Procedural operation.
- What if there are editing conflicts?
- Basic CVS commands.
- Useful aliases for remote operation.
- Advanced topics.



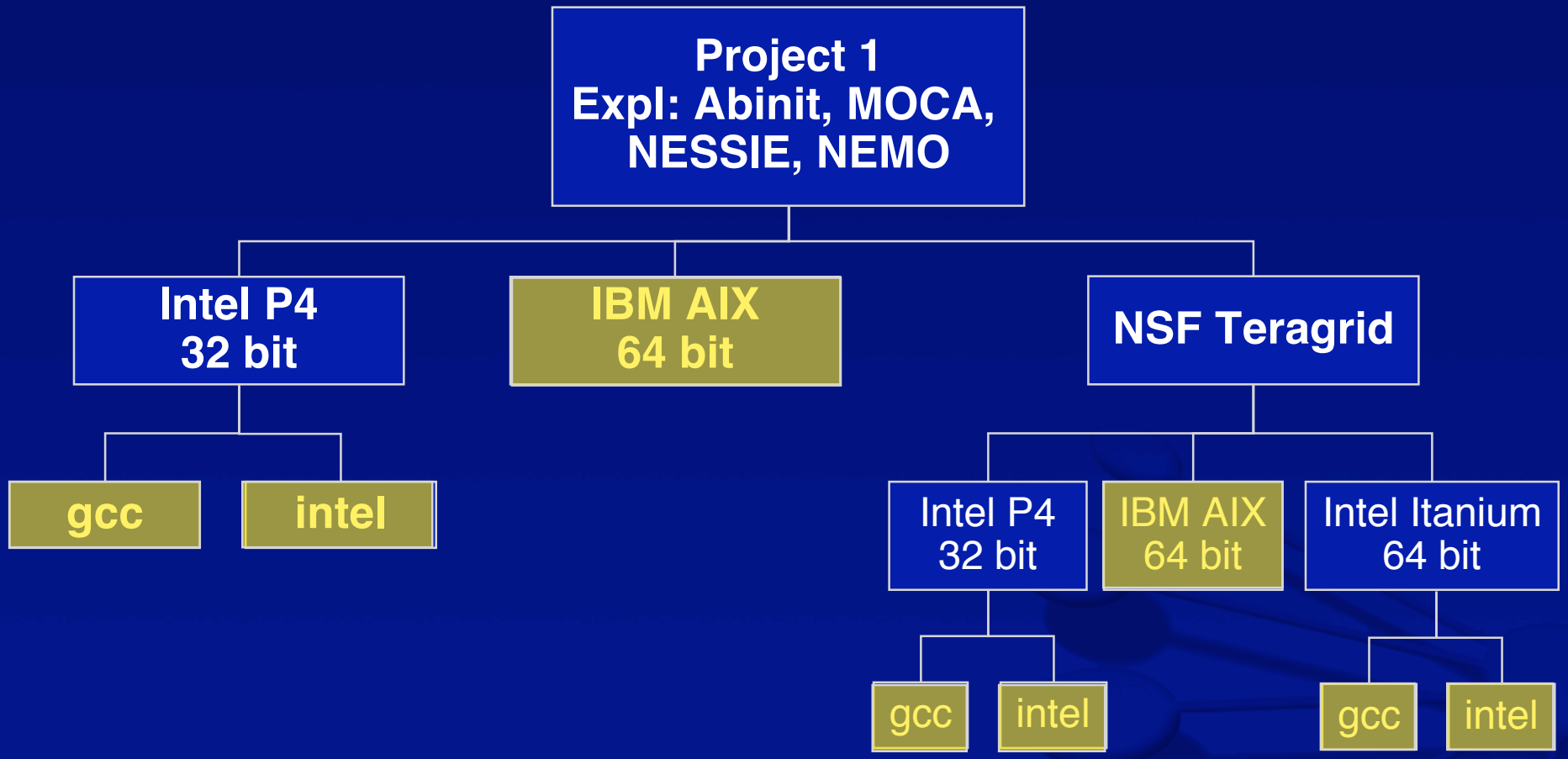
Who Should Care?

- Do you develop software in any language?
  - E.g. Matlab, C, Fortran, Python etc. - you are not just a user of e.g. MS Word or Gaussian?
  - Is your software rapidly evolving?
  - Do you keep “versions” of your code in different directories or on different machines?
  - Do you have trouble keeping track of the bugs you fixed for one project, but now you need it in another?
- Do you or should you be using multiple computer systems to get your work done?
- Do you or should you be developing software together with other people?

- If your answer is yes to any of these questions: Learn about CVS!

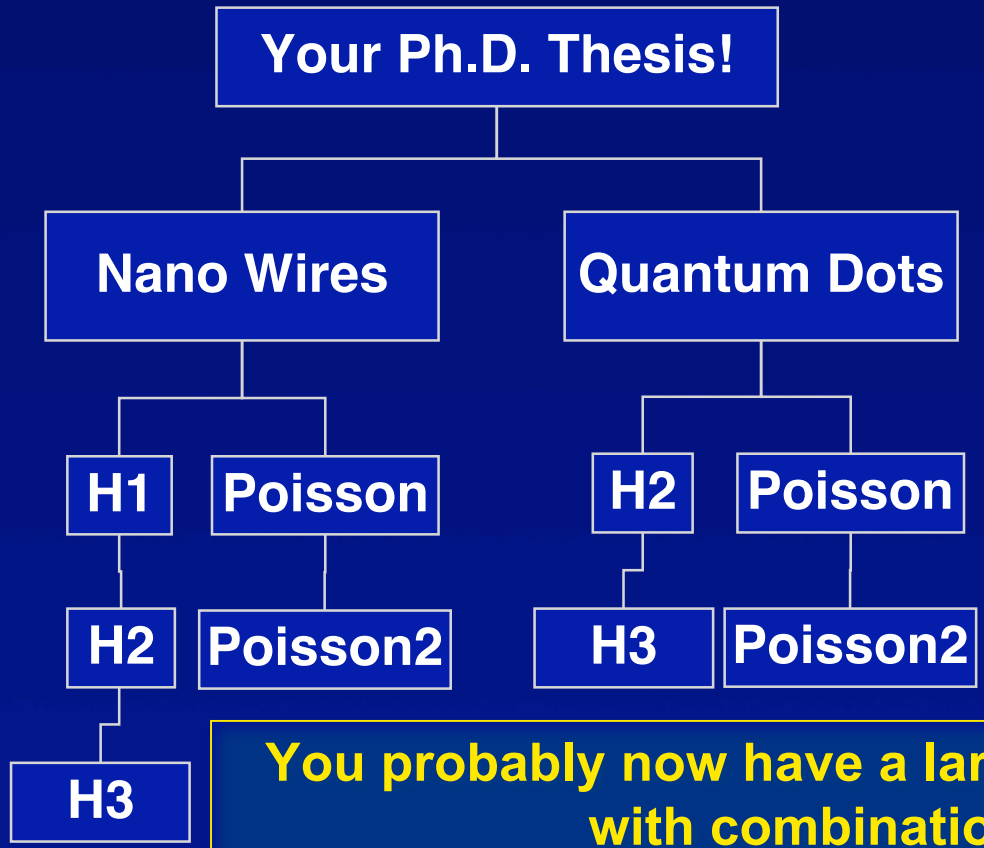


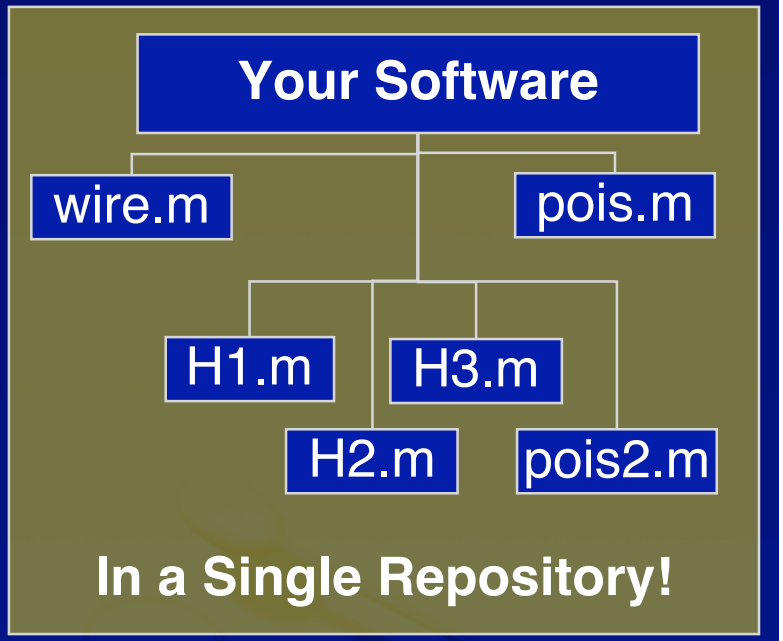
Have you worked on multiple computers with the "same" code?



Could end up with 8 different versions of the "same"code, with "slight" modifications!!!!!

Has your PhD lead to several versions of related pieces of code? Did you go through several iterations?





You probably now have a large number of directories with combinations of codes

Where do you keep all these versions of code?
What if someone asked you to give them your code?

CVS runs on top of RCS-Revision Control Systems

- Use a central library system.
  - Library contains files pertinent to a project.
  - Library resides on a server that can be connected to by remote machines.
  - File revisions automatically numbered
  - By default users see the latest version of a file.
  - Can retrieve any older version as well.
  - Can request information about the changes from one version to the next.

- Good Programming practice helps (but is not essential):
  - Break up the source code into modular files - modules are useful for many projects and people.
    - E.g: Computation of Hamiltonian matrix elements, vs. arrangements of matrix elements into geometry dependent matrix, vs. solution of the eigenvalues, vs. graphing, vs. data storage.
  - Eliminate duplication of code that needs to be maintained independently.

CVS
What are the basic commands?

User gekco:
edits file.c r1.3
he creates the new r1.4
commit = send to libr.

commit file.c

co file.c

Users lake, bowen, etc.:
look at the latest public revision of file.c (r1.3)
co = check-out

co file.c

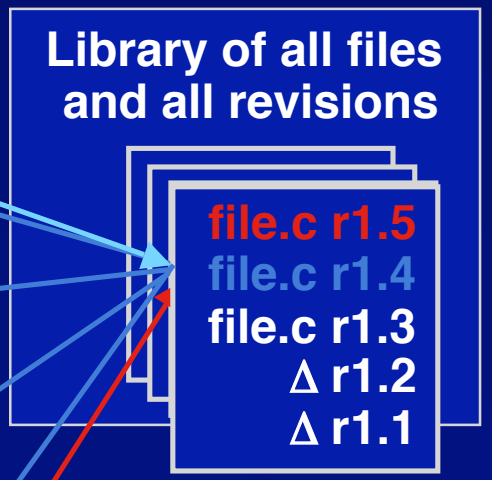
cvs update
or
co file.c

Users lake, bowen, etc.:
update to the latest public revision of file.c (r1.4)
co = check-out = update

cvs update
or
co file.c

Users mmc made changes to r1.3 but did not commit yet!
cvs update performs:
(r1.3+Δmmc) merge Δr1.4
cvs commit creates r1.5

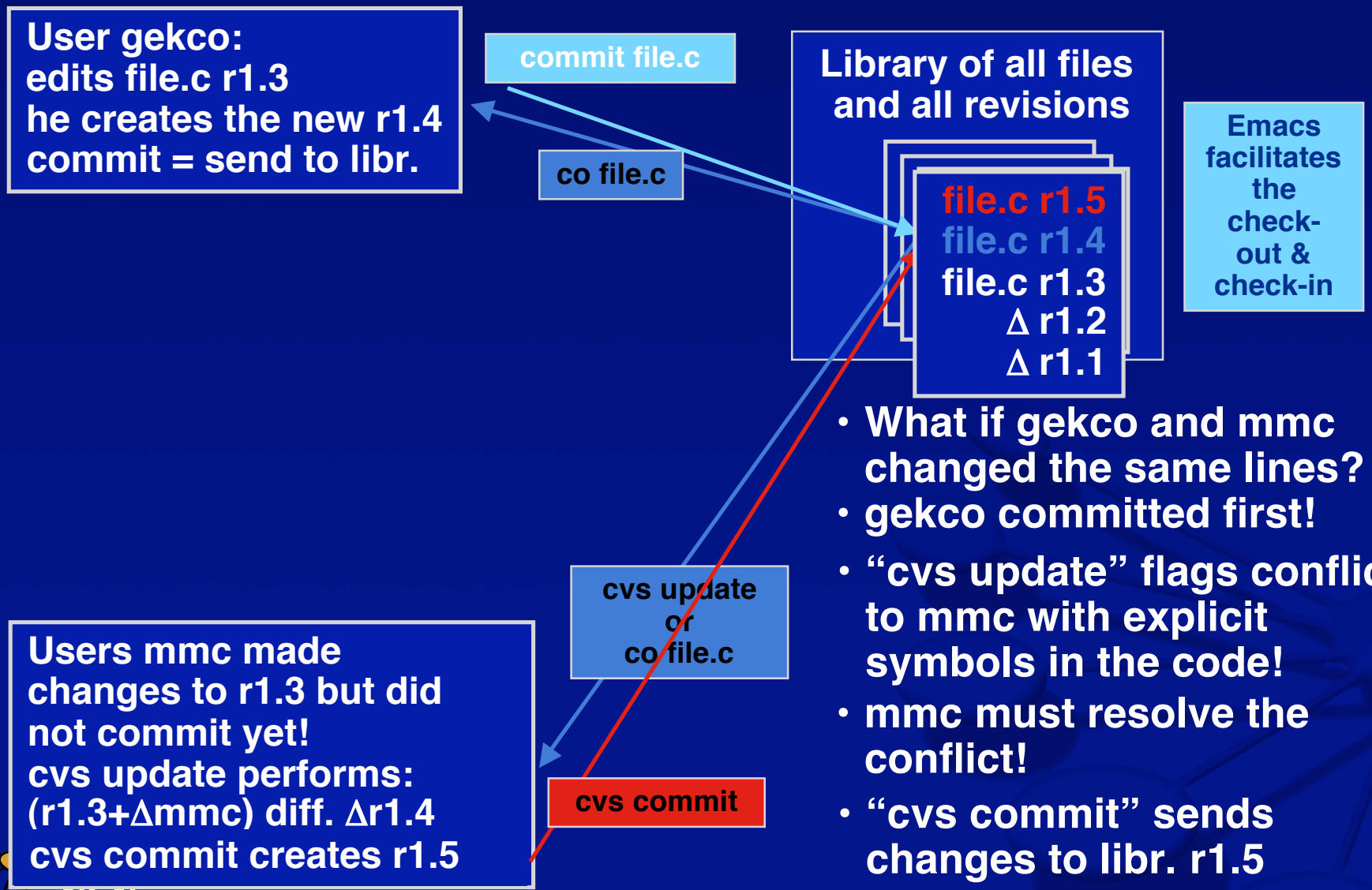
cvs commit



Emacs facilitates the check-out & check-in

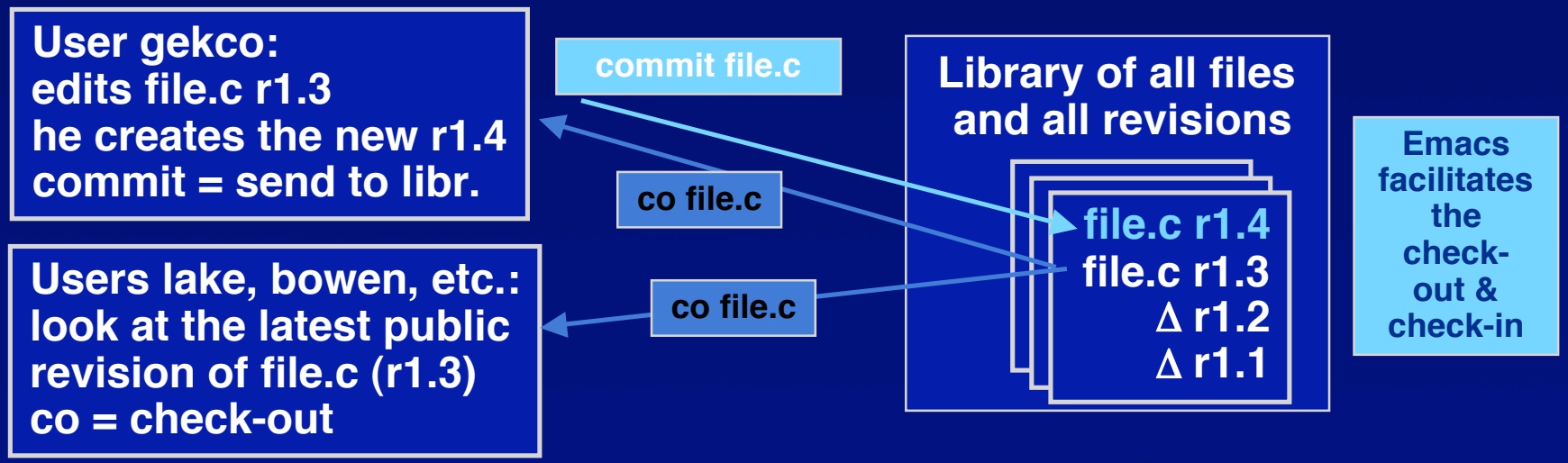
- mmc's changes are not lost
- "cvs update" performs update against latest copy and merges local changes
- "cvs commit" commits the local changes into the library

CVS
How are conflicts being handled?



- What if gekco and mmc changed the same lines?
- gekco committed first!
- "cvs update" flags conflicts to mmc with explicit symbols in the code!
- mmc must resolve the conflict!
- "cvs commit" sends changes to libr. r1.5

CVS - The Bottom Line



- CVS operates:
  - across remote platforms
  - Different platforms, Linux, IRIX, SunOS, AIX, Cygwin, Windows
- Emacs and other editors can help with the check-out and check-in!

FORGET about PRIVATE/LOCAL CODE VERSIONS!!!

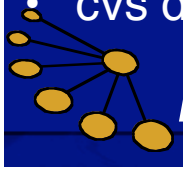
CVS - basic user commands

Assumption:CVS repository exists, a new member joins team

- "cvs co project" - retrieves a local copy of project in directory "project"
  - specification "where" the repository is on the next slide

All following commands work inside "dir_name" without reference to repository!

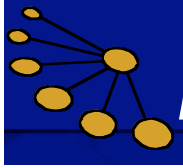
- "cvs update" - update the local copy against the latest copy in the library. Flag possible conflicts!
- "cvs commit <filename>" - commit local changes to the library. Implicitly performs "cvs update". If there is a conflict, "cvs commit" will fail and and demand conflict resolution. If a "filename" is given only that particular file will be committed. An editor window will pop up where the user should document the theme of the changes to the files.
- "cvs add filename" - add file called "filename" to the repository in the same relative directory tree. "filename" can be a directory name
- "cvs delete filename" - remove a file from the repository (it actually gets moved into an ATTIC directory).
- "cvs log <filename>" - prints the log of all the files in the directory tree or the log of "filename".
- "cvs diff <filename>" - prints out the differences to the local files or "filename" against the library files.



NCN

CVS - remote operation
Useful Environmental Variables

- Previous slide omitted the specification as to where to find the (remote) repository. Assume
  - "/some/pathname/repo" is the repository
  - "/some/pathname/repo/project1" is the particular project in the repository
  - "machine.domain" the machine where the repository resides.
  - "loginname" is the user's login on "machine.domain"
- There are several ways to specify the location of the repository.
  1) direct specification in the command line option:
     `cvs -d /some/pathname/repo co project1`
     will work on the local machine "machine.domain"
     `cvs -d :ext:loginname@machine.domain:/some/pathname/repo co project1`
     will work on a remote machine if "loginname" has password on "machine.domain"
  2) By specification of an environmental variable:
     `cvs co project1`
     will work if "`setenv CVSROOT "/some/pathname/repo`"
     or if "`setenv CVSROOT ":ext:login@machine.domain:/some/pathname/repo`"
     are set for local or remote access.
- Remote access typically requires the setting of a secure communication channnel:
  `setenv CVS_RSH "ssh"`



CVS - advanced topics

- **Adding a binary file to the repository:**
  - cvs add -kb filename
- **Creation of a CVS repository**
  - cvs init
  - **The location needs to be specified directly or through environmental variables as describes on the previous page.**
- **Import of a directory tree named "project2" into a repository**
  - cd project2
  - cvs import -m "some message" project2 VendorTag ReleaseTag "some message" **gets written into the log file of each file that is in** project2, VentorTag **is a single character string used to identify the origin of the code and** ReleaseTag **specifies the Vendor's release number**
  - cvs import -m "project2 import" project2 FromJoe 1 **should work fine for example**

- **More information on** http://www.gnu.org/software/cvs/manual/html_mono/cvs.html