CVS Tutorial

CSE 403 – Software Engineering

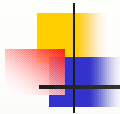**Konrad Lorincz**

Outline

- Creating a Repository
- Importing a Project
- Basic CVS Usage
- Branches in CVS
- Additional References

1

Creating the Repository

- **Important:** Create your CVS repository <u>in your group project workspace</u>. Your group project workspace is located on the instructional machines: tahiti, sumatra, ceylon

- When you want to check out a project copy, check it out in your personal account or in a subdirectory of the group project space.

- Your group project workspace is /projects/instr/02wi/cse403/cse403[a-n]

- For this example I will use groupa

- Here is what you should do

```
ssh sumatra.cs.washington.edu
cd /projects/instr/02wi/cse403/cse403a
```
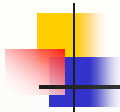
Creating a Repository

- Set the **CVSROOT** environment variable. (Note: in this example I am showing how to do this for 2 shells, csh and bash). You will probably want to add this to your shell login file.

```
setenv CVSROOT ~/cvsroot              [csh]
CVSROOT=~/cvsroot; export CVSROOT   [bash]
```

- Create a Repository

```
cvs init
```

  - will create the cvsroot directory with subdirectory CVSROOT

Importing a Project

- Assume we want to add `myproject` to CVS, which contains several files: `main.cpp`, `function1.cpp`, `function2.cpp`

  ```
  cd myproject
  cvs import -m "Importing MyProject" myproject
      group-a release-0
  ```

  - **"-m"** – specifies a log message (otherwise it starts an editor)
  - **group-a** – vendor tag
  - **release-0** – release tag

- Now we can delete the myproject directory
 ```
  rm –r myproject
  ```

---

# Basic CVS Usage

- Check out the source of your project.  In this example I will check it out in a subdirectory of the group project space. Here the name of the group member is Bob.

  ```
  cd /projects/instr/02wi/cse403/cse403a
  mkdir bob
  cvs checkout –d bob/myproject myproject
  ```

- This creates the `myproject` directory and puts the files: `main.cpp`, `function1.cpp`, `function2.cpp` and the subdirectory `CVS` which stores some information about the files
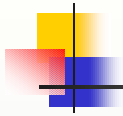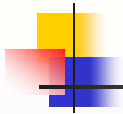
Basic CVS Usage

- Now we can edit our files. Let's add the following line in main.cpp

```
cout << "Second line in main()" << endl;
```

- Now we check in the new copy

```
cvs commit -m "Added a second print."
main.cpp
```

Basic CVS Usage

- Let's see the diff between two revisions

```
 cvs diff -r 1.1 -r 1.2 main.cpp
```

Index: main.cpp
===
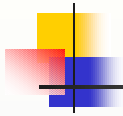*RCS file:*
retrieving revision 1.1
retrieving revision 1.2
diff -r1.1 -r1.2
9a10
>   cout << "Second line in main()" << endl;

Basic CVS Usage

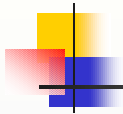- Diff with the Unidiff flag

```
cvs diff -u -r 1.1 -r 1.4 main.cpp

--- main.cpp    2002/01/17 08:47:00    1.1
+++ main.cpp    2002/01/17 09:27:18    1.4
@@ -6,7 +6,8 @@


 int main() {
-       cout << "First line in main()" << endl;
+       cout << "Second line in main()" << endl;
+       cout << "Third line in main()" << endl;

        function1();
        function2();
```
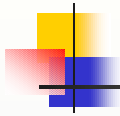
Branches in CVS

- CVS allows to branch on the revision tree. Allows you to maintain several releases of the software (note: this is different from versions).

```
cvs tag release-1
```

- At a later point in time we could check out release-1

```
cvs checkout -r release-1 myproject
```

Branches in CVS

- Let's say you discover a serious bug in release-1, while you are well into release-2. You need to fix release-1 for you customer. So, you create a branch of release-1.

  ```
  cvs rtag -b -r release-1 release-1-patch myproject
  ```

- Now we create a working copy of the branch

  ```
  cvs checkout -r release-1-patch myproject
  ```

Additional References

- A simple tutorial
  http://www.cs.duke.edu/~ola/courses/tools/cvs.html

- CVS Homepage
  http://www.cvshome.org
- CVS Manual
  http://www.cvshome.org/docs/manual

- Windows version (WinCVS)
  http://www.wincvs.org
- Other Windows versions
  tkCVS, jCVS