This short how-to is meant to be a crash course in using CVS within Eclipse in the CS labs at MTU. It outlines the general information that users will need to know to ensure their project is kept in a source code versioning system called CVS.

1. CVS

1.1. CVS?

CVS is the preferred method of source code revisioning that is used in the Computer Science Labs at Michigan Tech. It is a very powerful system that is widely used. It allows users to keep many different versions of their code in one place for easy access. It will allow a user to go back through revisions and track changes. CVS will also allow a group to work on a project concurrently as well as allow a group to communicate changes easily with regard to source code updates.

1.2. Why CVS?

Many programmers choose to use CVS to make sure they can track changes to a given project. Also, many programmers set up CVS repositories so they can add other programmers to a project and have a central location of all source code that is being worked on. CVS will also allow a group to keep one central, current version of an application that is known-working. It helps users of the repository to roll back to a previous version of the code if their current version becomes non-functional.
2. Setting up CVS

Before CVS can be used, a location for repositories will need to be setup and initialized. A good location for this would be the root of your home directory (~/). It is generally good practice to name this directory CVS so that you know what is within the folder. The folder does not have to reside within the root of your home directory or be named CVS, but it is generally a good idea to help simplify more advanced CVS tasks.

To create the CVS repository location and initialize it, follow the procedure below.

**Note**
This directory can be used for multiple projects at the same time so there is no need to create an independant folder for each project you have. However, there can be issues regarding space utilization if you do not clean files out of this directory when projects are completed. See Section 6, “Things to Keep in Mind” for more information regarding this.

**Warning**
Do NOT run the following procedure when working in a group environment more than once. If you run the following commands more than once you may accidentally delete files already checked into the CVS repository or render the repository unusable.

1. Open a terminal
2. Create ~/CVS using **mkdir**
3. Run **setenv CVSROOT /major/userid/CVS**

**Warning**
This procedure is different if you are running Bash instead of the default shell in the CS labs. If you are running Bash, the command should be **export CVSROOT=/major/userid/CVS**.

**Note**
The userid variable in the above command should be replaced with your userid.

4. Run **cvs init**

After you run the above commands you should be ready to use CVS with Eclipse, Gvim, Kate or any of the other editors available in the labs. Only Eclipse will be covered in this guide. However, if you want to use CVS with other editors, look at the CVS guide online. A link to the appropriate information is included in Appendix A, *Links to external information*.

3. Using CVS within Eclipse

3.1. Initial Setup

To setup Eclipse to work with your CVS repository follow the procedure below.

**Note**
This procedure will allow you to work on code from home as well as in the labs. The only potential problem is a difference between the Java Runtime or C/C++ compiler you are using at home.
1. Create your new project and copy files that you need using the appropriate scripts (CS1121) or create your initial source files (all other classes). You can use Eclipse or the command line for creating and copying files.

**Note**
If you use the command line, you will need to refresh the directory listing in Eclipse by right-clicking the project name and selecting Refresh.

2. Switch your perspective to **CVS Repository Exploring**. It should look something like what is shown below.

**Note**
This can be done by clicking on the icon with the addition symbol in the upper right corner.

**Figure 1. The default CVS perspective**

3. Right-click in the navigator window (pane on left).
4. Choose New -> Repository Location.
5. In the dialogue that comes up, set the following values:

**Note**
If you are having trouble finding certain fields, refer to the screenshot below the list.

- **Host**: wo.pr.csl.mtu.edu
• Repository Path: /major/your_userid/CVS
• User: your_userid
• Password: your_password
• Connection Type: extssh
• Save Password: Check if you want your password saved
• All other options: Use the default value

Figure 2. Eclipse CVS Setup
Note
The above procedure will work with Windows as well. If you are working from a Windows PC you will not have to perform any additional configuration after installing Eclipse.

3.2. Adding a project to CVS

After you make Eclipse aware that a CVS repository is available for usage (the steps above), you need to add your project to the CVS repository. Follow the procedure below to do this.
1. Switch perspectives so that you are back to the perspective associated with your project (the C/C++ perspective for C or C++ projects and Java for Java projects).

2. Right-click your project in the Navigator window on the left and choose Team -> Share Project.

**Note**
You need to select Team -> Share Project because adding a project to CVS is similar to sharing a project for a team. If you want the project to be accessible by only your user account, you will need to set the permissions within the CVS directory to 700.

3. Make sure the radio button for existing repository is selected and choose the repository you created above and click Next.

**Figure 3. Eclipse's Share Project Screen (1)**

![Eclipse's Share Project Screen](image)
4. Select "Use project name as module name" and click Next.

**Figure 4. Eclipse's Share Project Screen (2)**

![Eclipse's Share Project Screen (2)](image)

5. Click Next. The dialogue you are clicking through is for more advanced purposes when working with large code repositories.

**Figure 5. Eclipse's Share Project Screen (3)**
6. Click Finish.

**Figure 6. Eclipse's Share Project Screen (4)**
3.3. Versioning your code

Eclipse includes a Team Synchronizing perspective that allows you to check in and update code that is in a repository. To check in new versions of code, follow the procedure below.

1. Switch to the Team Synchronizing perspective. It should look similar to what is shown below.

   Figure 7. Team Synchronizing Perspective
2. Right-click your project in the Synchronization window (on the left) and select Commit.

3. A dialogue should pop up that looks similar to the one below. Fill in the comment field and make sure all files you want to update are selected.

Figure 8. Commit Files Dialogue
4. Click Finish.

**Important**

If you are checking binary files in to the repository, the dialogue box will look like the following figure instead of the normal dialogue.

**Figure 9. Binary Files Commit Dialogue**
4. Completing a project

4.1. Submit

After you finish writing your program, you will need to submit it for grading. In the CS department, a special program called `submit` will be used to submit your source code. Most professors prefer that you use the command line form of the program, but there is a GUI available when running `submit` without any arguments. If you need help, the lab consultants can assist you.

4.2. Unsharing Your Project

After completing your project you will need to unshare the project within Eclipse. To do this follow the
procedure below.

1. Right-click the project in the Eclipse Navigator window.
2. Click on Team.
3. Click on Disconnect.
4. In the dialogue that comes up, select whether or not to delete the CVS meta information by choosing the appropriate radio button.

**Note**
I would recommend deleting the meta information since the project is going to be removed from CVS.

![Figure 10. Disconnecting From a Project](image)

5. Click on Yes.

### 4.3. CVS

After you finish your project, the easiest way to clean up some of the space used by CVS is to remove the directory you initially created at the beginning of this guide. However, if you have more than one project in your repository, you can remove solely the project you finished quite easily. To remove only the current project from CVS while leaving the rest of the CVS directory, follow the procedure below.

1. Open a terminal and change your current directory to the root CVS directory (the one used when running `cvs init` in Section 2, “Setting up CVS”).
2. Run `ls` and look for the name of the project you were working on.
3. After finding the proper directory, run `rm -r project_name` and say yes or no to any of the warnings that come up accordingly.
4. Exit your terminal.
After you run the above procedure, your project should no longer exist in CVS. It will however remain stored in your home directory. It will not be removed from the working directory Eclipse uses, only from CVS.

5. Working in Group Environments

Many classes offered at Michigan Tech will allow a group of students to work together on a project. Eclipse streamlines this process by providing users a Team Synchronization perspective to work in. This perspective will show you changes that you need to add to the repository, changes that you need from the repository and changes that conflict with each other. Below are details about how to work within this perspective.

**Warning**

When working with a group, make sure there is only one central CVS repository and that all users have the proper permissions set on the files in the repository. The permissions in the CVS directory should be set to 770 for the folder with your project name and all of the files within the folder.

When working with a group one of the group members should initialize the CVS repository using the procedure given in Section 2, “Setting up CVS” and add the initial source files to the repository. After completing these tasks group members will need to check out the project from CVS using the procedure below.

1. Add a repository following the instructions in Section 3.1, “Initial Setup”.
2. Open the newly added repository by clicking the arrow next to the name.
3. Open the HEAD item underneath the repository name by clicking the right arrow next to HEAD.

**Note**

After completing the above three steps Eclipse should look similar to the following:

**Figure 11. Finding a Project to Checkout**
4. Right click the project name in the Navigator window and select **Check Out As**.

5. In the dialogue that comes up select **Finish**.

   **Figure 12. Checking out a Group Project**
6. In the dialogue that comes up, select the type of project you are checking out.

**Note**

The following is a list of languages and their associated project types.

- C: "Standard Make C Project"
- C++: "Standard make C++ Project"
- Java: "Java Project"

**Figure 13. Selecting a Project Type**
7. After selecting the project type, click **Next**.

8. Fill in the project's name in the next screen and select **Finish**

   **Figure 14. The default CVS perspective**
9. A new dialogue may pop up asking if you would like to switch perspectives. Select **Yes** to open the perspective associated with the project.

**Figure 15. The default CVS perspective**
After completing the above procedure group members will be able to check in and update code as outlined in Section 3.3, “Versioning your code”.

Note

This guide cannot cover the necessary information that is required to fully utilize the Team Synchronization Perspective. Please refer to the Eclipse [http://eclipse.org](http://eclipse.org) website for more information.

6. Things to Keep in Mind

6.1. Reverting to the Repository Version

If you are working on your code and realize you need to go back to the last checked in version of your program, you can. In order to revert to the last version of your program that is in CVS, you will need to open the Team Synchronizing perspective. After opening the Team Synchronizing perspective you should right-click the file you want to revert and select Revert. After selecting Revert, your copy of the source file should be reverted to the last version that was checked into CVS.

6.2. Permissions

6.2.1. Group Work

When working with a group you will need to keep permissions in mind. Since there is no centralized CVS server for the CS department, users are required to run CVS locally through their user account and home directory. If members of the group have not been granted permissions to the proper areas of the CVS repository they will not be able to check changes in or out correctly. Make sure the permissions on the CVS directory for your project are set to 770 and all files within the directory are set to 770 as well.

6.2.2. Individual Work

When working independantly, it is important to remember to properly set permissions on your CVS repository and working copy of your program. If you have improper permissions set, another user could go through your home directory and copy your code from your home directory to their home directory. If you set the permissions of your repository to 700, only you should be able to read and write the necessary files.

6.3. Space Utilization

CVS keeps a copy of every version of your program that you check in. This feature of CVS will become a problem when you have many projects currently versioned in the repository because your home directory has a limited amount of storage space. If you do not clean out old code versions after turning them in, you will quickly fill your home directory.

See Section 4.3, “CVS” for more information on how to remove completed programs from CVS.

6.4. Binary Files

CVS is infamous for its handling of binary files. If possible, avoid adding binary files to your CVS repository. There is information available online regarding this issue if you would like more details.
A. Links to external information

A few websites that were used to compile this guide have been included.

- A great guide on using CVS for your projects:  http://cvsbook.red-bean.com/

B. Other Perspectives

- There are many other perspectives available for Eclipse that you may find useful. Below is a list of common perspectives that are available.
  - PyDev, A Python environment for Eclipse:  http://pydev.sourceforge.net/