Experiment No: 10

● **Aim: Version Controlling of the project.**

● **Theory:**

● **Project Name: - The QR CODE SCANNER**

Version control system:

A version control system is a software that tracks changes to a file or set of files over time so that you can recall specific versions later. It also allows you to work together with other programmers. The version control system is a collection of software tools that help a team to manage changes in a source code. It uses a special kind of database to keep track of every modification to the code. Developers can compare earlier versions of the code with an older version to fix the mistakes.

Benefits of the Version Control System

The Version Control System is very helpful and beneficial in software development; developing software without using version control is unsafe. It provides backups for uncertainty. Version control systems offer a speedy interface to developers. It also allows software teams to preserve efficiency and agility according to the team scales to include more developers.

Some key benefits of having a version control system are as follows.

- Complete change history of the file
- Simultaneously working
- Branching and merging
- Traceability

repository – single location where the current and all prior versions of the files are stored

working copy – the local copy of a file from the repository which can be modified and then checked in or “committed” to the repository

check-out – the process of creating a working copy from the repository (either the current version or an earlier version)

check-in – a check-in or commit occurs when changes made to a working copy are merged into the repository

diff – a summary of the differences between a working copy and a file in the repository, often taking the form of the two files side-by-side with differences highlighted

conflict – a conflict occurs when two or more developers attempt to make changes to the

same file and the system is unable to reconcile the changes (note: conflicts generally must be resolved by either choosing one version over the other or by integrating the changes from both into the repository by hand)

update – merges recent changes to the repository into a working copy

Types of Version Control System:

- Localized version Control System
- Centralized version control systems
- Distributed version control systems

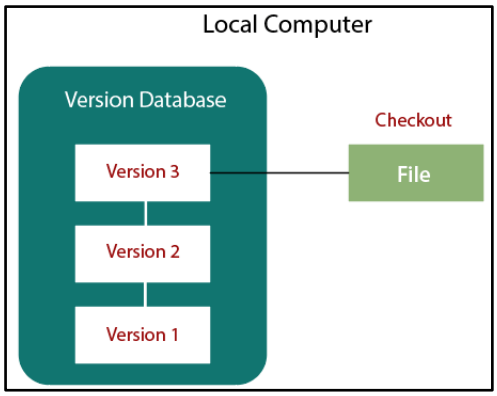
localized version control:

The localized version control method is a common approach because of its simplicity.

But this approach leads to a higher chance of error. In this approach, you may forget

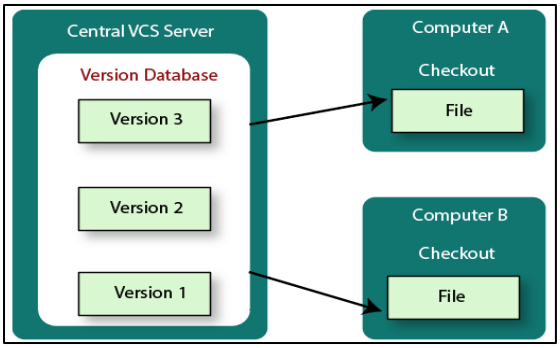
which directory you're in and accidentally write to the wrong file or copy over files you

don want to.



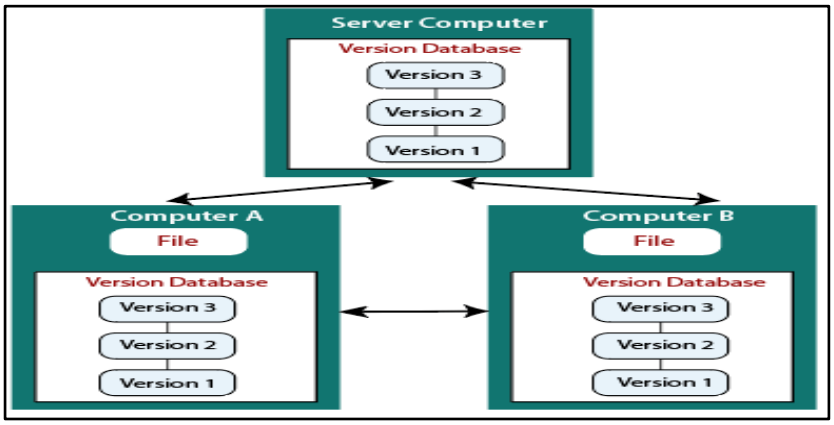
Central version control:

The developers needed to collaborate with other developers on other systems. The localized version control system failed in this case. To deal with this problem, Centralized Version Control Systems were developed.



Distributed version control:

In a Distributed Version Control System (such as Git, Mercurial, Bazaar or Darcs), the user has a local copy of a repository. So, the clients it just checks out the latest snapshot of the files even they can fully mirror the repository. The local repository contains all the files and metadata present in the main repository.

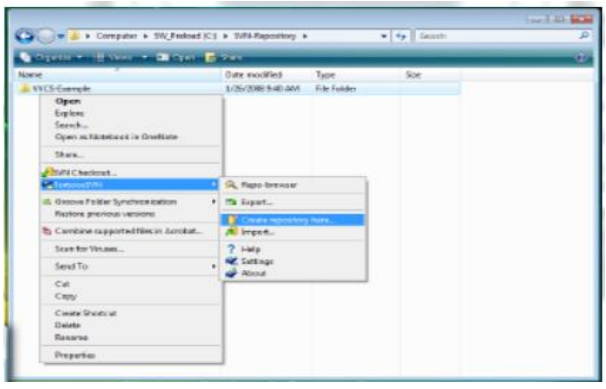


The basic steps that one would use to get started with a **version control tool** are as follows:

1. Create a repository
2. Import a directory structure and/or files into the repository
3. Check-out the repository version as a working copy
4. Edit/modify the files in the working copy and examine the differences
5. between the working copy and the repository (i.e., diff)
6. Check-in (or commit) the changes to the repository.

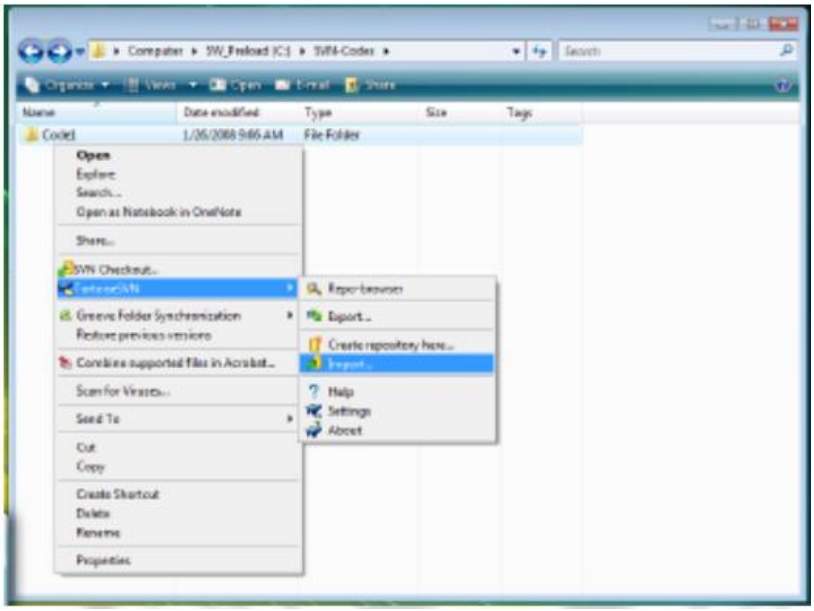
1. Creating a Repository:

Determine a location for the repository, ideally on a server which is automatically backed up. Create a folder with the name of the repository; in this example the repository is called “VVCSExample.” Right click on the folder name, choose “TortoiseSVN” (which is integrated into the Microsoft Windows Explorer menu), then “Create Repository Here.” Choose the Native Filesystem, then you should see the message “Repository Successfully Created.



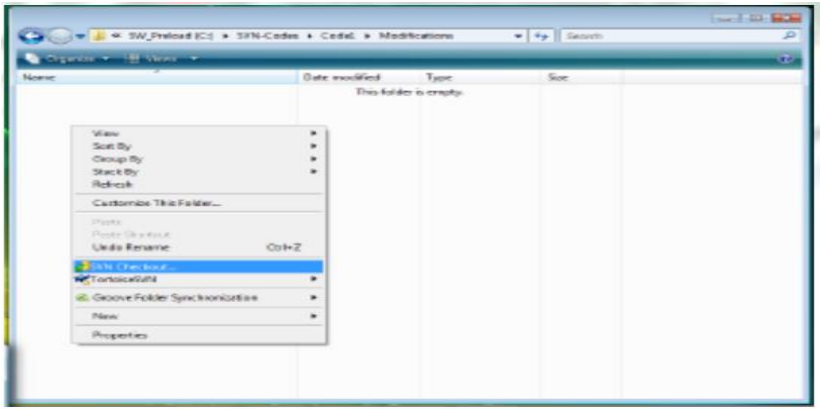
2. Importing a File into the Repository

Right click on the directory containing the file(s) and/or directory structure you wish to import to the repository (note, the directory that you click on will not be imported). Here we will simply be importing the file “code1.f” from directory “Code1.” This code creates a 17×17 two-dimensional Cartesian grid for x and y between 0 and 1. Browse until you find the location of the repository “VVCS-Example” and select that directory name. This version of the code will be Revision 1.



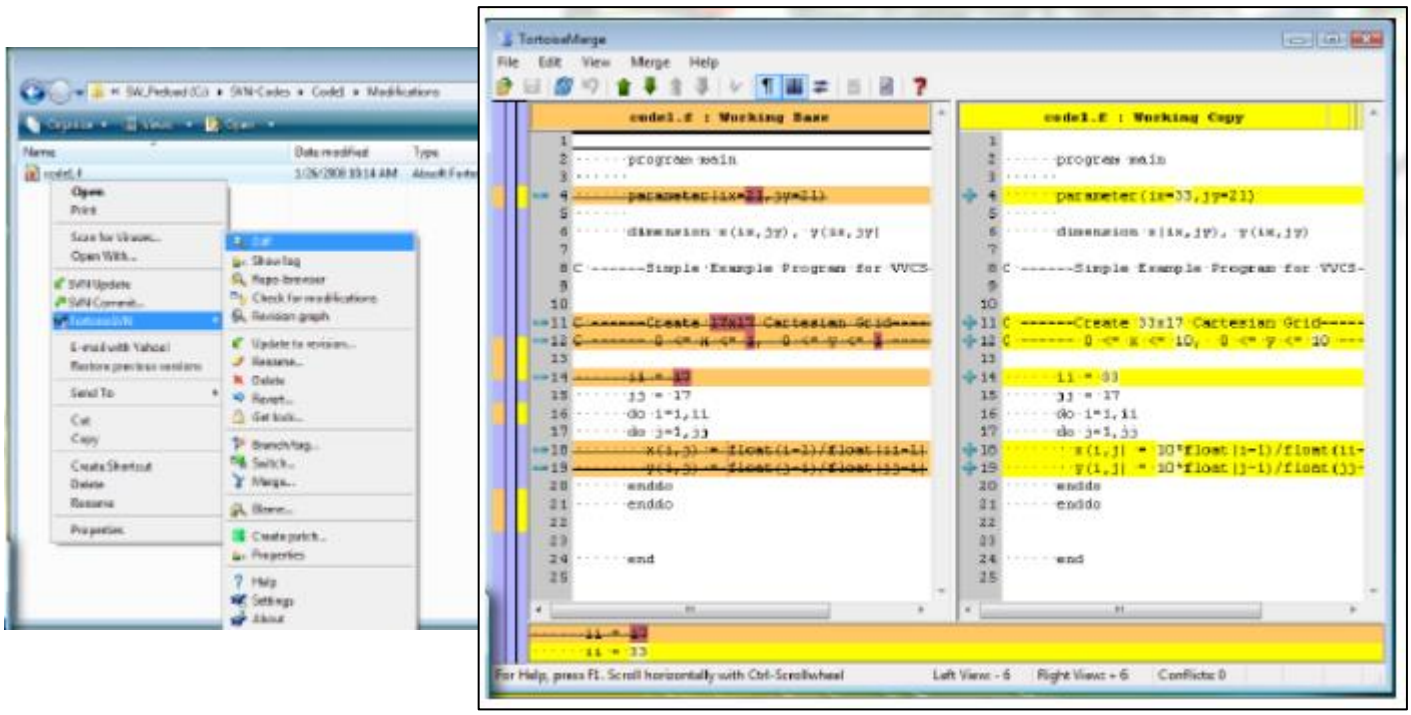
3. Checking the Code out from the Repository

You now have the code “code1.f” safely placed in the repository. To modify this code and create a new revision, you will need to check out a working copy of the code. Go to the directory where you will be modifying the code, in this example, the directory “Modifications.” Right click in Windows Explorer, and select “SVN Checkout…” Select the name of the repository you just created, then click “OK.” You will now get a window telling you that you are at Revision 1. Notice the green check mark on the “code1.f” icon. This indicates that this working copy is up to date with the version in the repository.



4. Modify the Code and Compare to the Repository Version

The code "code1.f" can now be modified. Here we will change the code to allow the Cartesian grid to contain 33x17 points between the values of zero and ten. Once the code has been modified, you will notice that the green check mark has been replaced by a red exclamation point, indicating that the current working copy has been modified from the version in the repository. To examine these differences, right click on the "code1.f" file, select "TortoiseSVN, "then "Diff." This opens the "Tortoise Merge" tool which clearly shows the modifications to the repository version (Working Base) that were made in the Working Copy.



● **Conclusion:** We, understood the basics of **Version Control** and the **benefits** of **Version Control**. Now, you know why this system is so widely used and why small or big company opts for Version Control