KING ABDULAZIZ UNIVERSITY
THE COLLEGE OF ENGINEERING

OPERATING SYSTEMS

EE463

LAB ASSIGNMENT #1: VERSION CONTROL SYSTEM

STUDENT NAME: SAMER SHOUKAT KHAN

ID: 1306219

SUBMISSION DATE: 09/05/2015

SUBMITTED TO

ENG. TURKI GARI

What is version control?

Version control is the art of managing changes to information. It has long been a critical tool for programmers, who typically spend their time making small changes to software and then undoing those changes the next day. But the usefulness of version control software extends far beyond the bounds of the software development world. Anywhere you can find people using computers to manage information that changes often, there is room for version control.

Centralized vs. Decentralized

Early VCSes were designed around a centralized model in which each project has only one repository used by all developers. All “document management systems” are designed this way. This model has two important problems. One is that a single repository is a single point of failure — if the repository server is down all work stops. The other is that you need to be connected live to the server to do checkins and checkouts; if you're offline, you can't work.

The very earliest first-generation VCSes supported local access only; that is, all developers of a project needed to be on the same machine as the single central project repository. Second-generation VCSes, while still centralizing each project around a single repository, supported a client-server model allowing developers to work with it over a network from other machines.

Newer, third-generation VCSes are decentralized. A project may have several different repositories, and these systems support a sort of super-merge between repositories that tries to reconcile their change histories. At the limit, each developer has his/her own repository, and repository merges replace checkin/commit operations as a way of passing code between developers.

An important practical benefit is that such systems support *disconnected operation*; you don't need to be on the Internet to commit to the repository because you carry your own repository around with you. Pushing changesets to someone else's repository is a slower but also less frequent operation.

The Repository

At the core of the VCS is a repository, which is the central store of that system's data. The repository usually stores information in the form of a *filesystem tree*—a hierarchy of files and directories. Any number of *clients* can connect to the repository, and then read or write to these files. By writing data, a client makes the information available to others; by reading data, the client receives information from others. As the files in the repository are changed, the repository remembers each version of those files.

Tracking changes

You specify a directory or set of files that should have their changes tracked by version control. This can happen by checking out (or cloning) a repository from a host, or by telling the software which of your files you wish to have under version control. The set of files or directories that are under version control are more commonly called a repository, as mentioned above. As you make changes, it will track each change behind the scenes. The process will be transparent to you until you are ready to commit those changes.

Committing

Instead of recording each change individually, the version control system will wait for you to submit your changes as a single collection of actions. In version control, this collection of actions is known as a commit.

Conflicts

A conflict occurs when your changes are so similar to the changes that another team member made that the VCS cannot determine which the correct and authoritative change is. The VCS will provide a way to view the difference between the conflicting versions, allowing you to make a choice. You can either edit the files manually to merge the options, or allow one revision to win over the other.

Getting Updates

Getting the latest changes from a repository is as simple as doing a pull or update from another computer (usually a hosted or centralized server). When an update or pull is requested, only the changes since your last request are downloaded.

Diffing (or, viewing the differences)

By viewing a diff, you can compare two files or even a set of files to see what lines of code changed, when it changed and who changed it. Most version control tools let you compare two sequential revisions or two revisions from anywhere in the history.

Branching and merging

A branch allows you to create a copy (or snapshot) of the repository that you can modify in parallel without altering the main set. You can continue to commit new changes to the branch as you work, while others commit to the trunk or master without the changes affecting each other. Once you're comfortable with the experimental code, you will want to make it part of the trunk or master again. This is where merging comes in. Since the version control system has recorded every change so far, it knows how each file has been altered. By merging the branch with the trunk or master (or even another branch), your version control tool will attempt to seamlessly merge each file and line of code automatically. Once a branch is merged it then updates the trunk or master with the latest files.

Examples

Version control is provided at sites such as Github, SourceForge and Google Code. These sites typically build a suite of services around version control: archiving, release downloads, mailing lists, bug trackers, web hosting and build farms. This range of functionality makes them particularly attractive for those projects that do not have the resources to maintain their own server for version control. CVS used to be the most widely used open source version control system but these days Subversion and Git have overtaken it as most commonly used in open source projects. The basic capabilities of these systems are very similar, but they offer different security, networking and abstraction functionality, and different licenses

REFERENCES

1. Daityari, Shaumik. "Version Control Software in 2014: What Are Your Options?" Sharing Our Passion for Building Incredible Internet Things. Web. 8 May 2015.
2. "OSS Watch Provides Unbiased Advice and Guidance on the Use, Development, and Licensing of Free Software, Open Source Software, and Open Source Hardware." What Is Version Control? Why Is It Important for Due Diligence? N.p., n.d. Web. 08 May 2015. http://oss-watch.ac.uk/resources/versioncontrol
3. "Understanding Version-Control Systems (DRAFT)." Understanding Version-Control Systems (DRAFT). N.p., n.d. Web. 08 May 2015. http://www.catb.org/esr/writings/version-control/version-control.html#centralized_vs_decentralized
4. "What Is Version Control: Diffs and Patches - Atlassian Blogs." N.p., 09 Feb. 2012. Web. 08 May 2015. http://blogs.atlassian.com/2012/02/version-control-diffs-patches/?utm_source=wac-dvcs&utm_medium=text&utm_content=what-is-version-control
5. "Beanstalk Guides." An Introduction to Version Control • . N.p., n.d. Web. 08 May 2015. http://guides.beanstalkapp.com/version-control/intro-to-version-control.html
6. "Ubuntu Documentation." Version Control System. N.p., n.d. Web. 08 May 2015. https://help.ubuntu.com/lts/serverguide/version-control-system.html