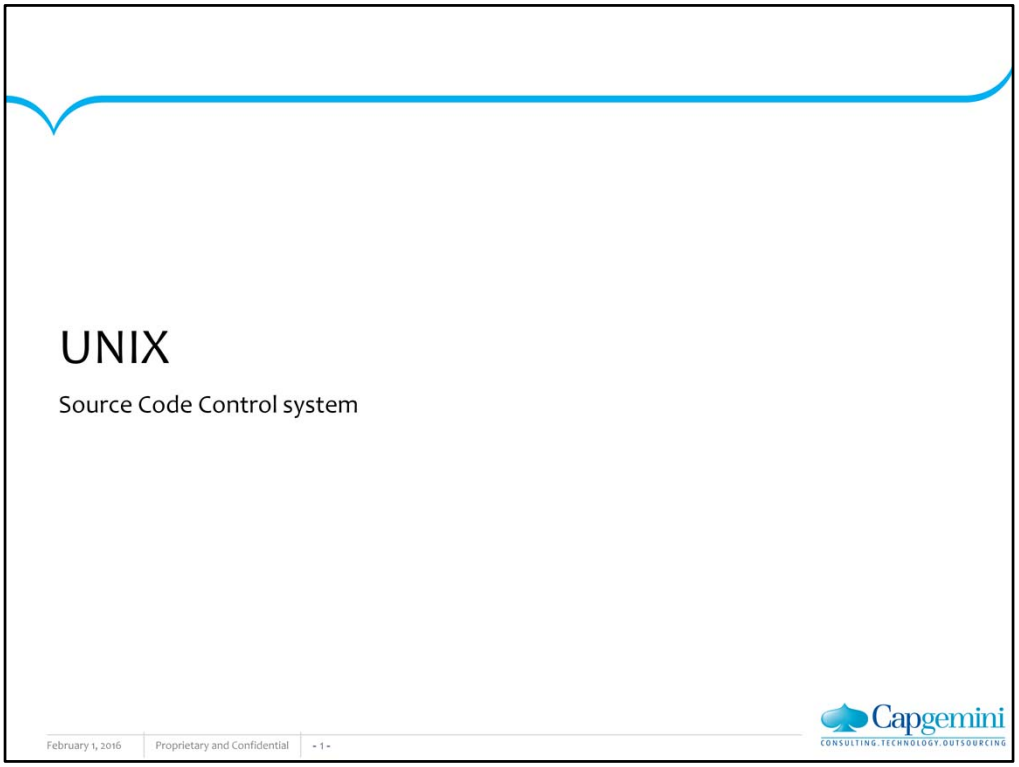
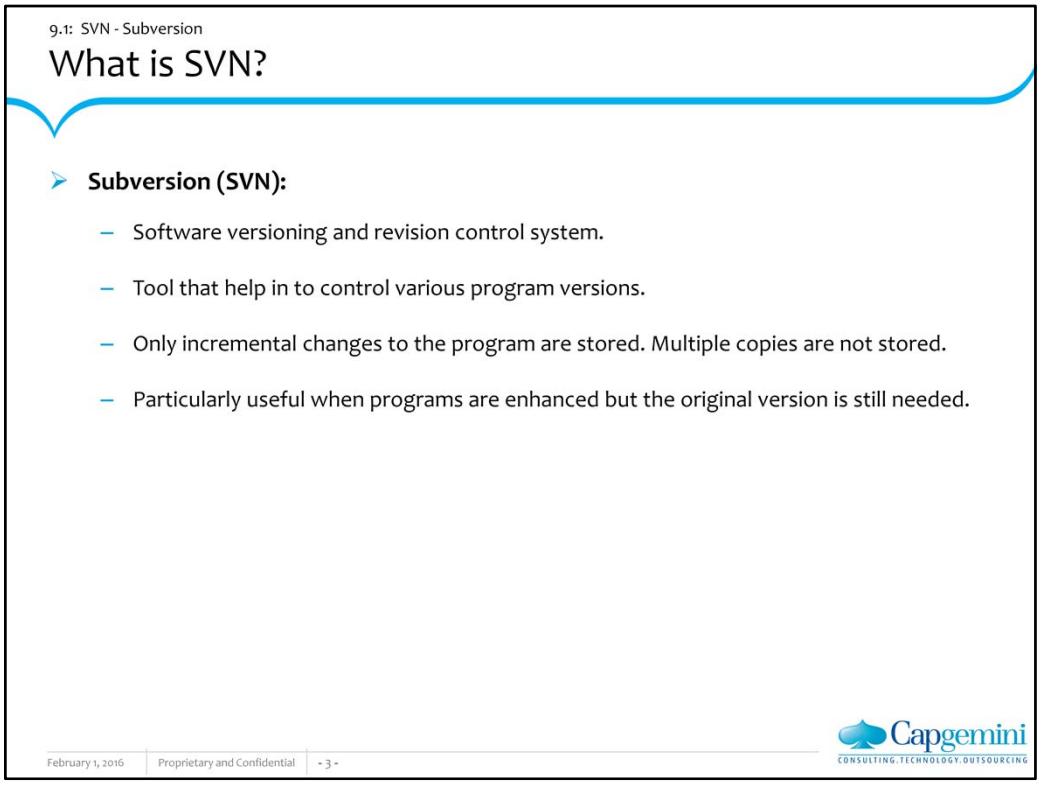

UNIX
Source Code Control system
February 1, 2016
Proprietary and Confidential
- 1 -
Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

## Lesson Objectives

- SVN




Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

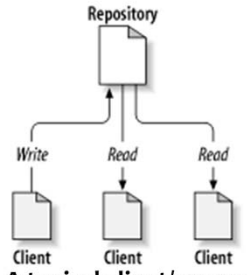



A version control system (or revision control system) is a system that tracks incremental versions (or revisions) of files and, in some cases, directories over time. Of course, merely tracking the various versions of a user's (or group of users') files and directories isn't very interesting in itself. What makes a version control system useful is the fact that it allows user to explore the changes which resulted in each of those versions and facilitates the arbitrary recall of the same.

9.1: SVN - Subversion

## Repository

- Repository is the central store of that system's data.
- The repository usually stores information in the form of a file system tree ( a hierarchy of files and directories)
- Any number of clients connect to the repository, and then read or write to these files.




A typical client/server



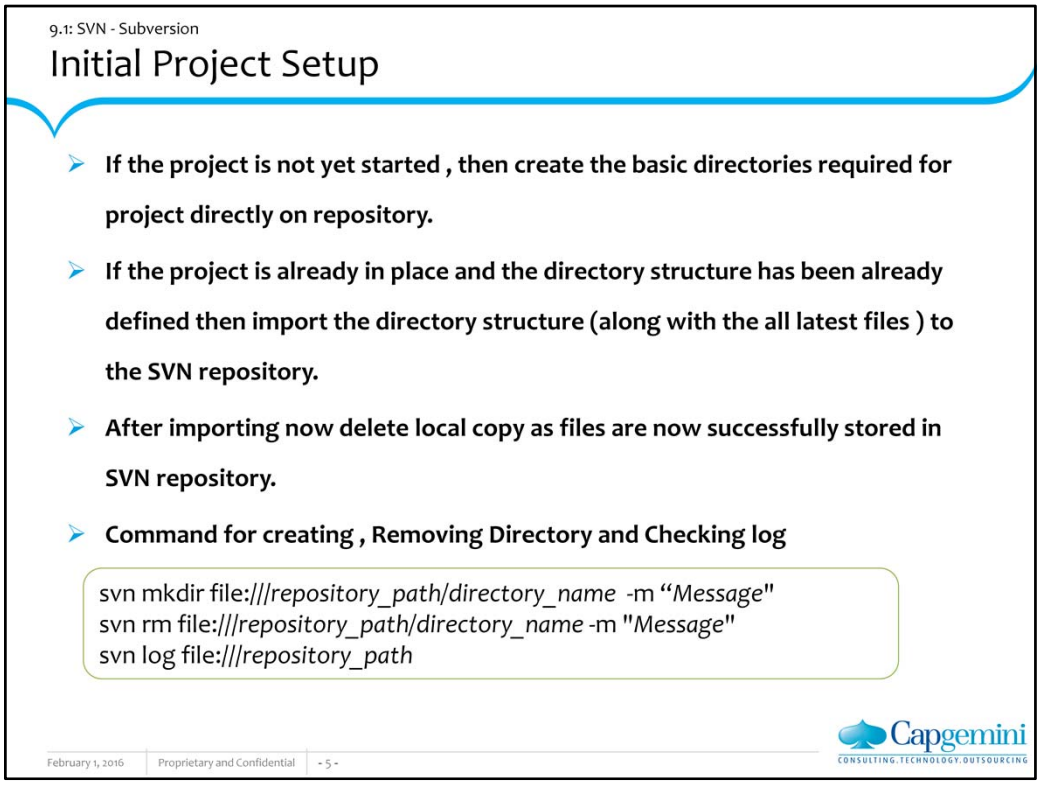
Capgemini CONSULTING.TECHNOLOGY.OUTSOURCING

At the core of the version control system is a repository, which is the central store of that system's data. The repository usually stores information in the form of a files system tree -- a hierarchy of files and directories. Any number of clients 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.

What makes the repository special is that as the files in the repository are changed, the repository remembers each version of those files.

When a client reads data from the repository, it normally sees only the latest version of the files system tree. But what makes a version control client interesting is that it also has the ability to request previous states of the files system from the repository. A version control client can ask historical questions such as What did this directory contain last Wednesday? and Who was the last person to change this file, and what changes did he make? These are the sorts of questions that are at the heart of any version control system.

**Note** : Repository creation is a one time activity which happens on server and usually done by system administrator.




**Note** : Initial Project Setup creation is a one time activity which can be done by administrator or user .

Following is the command to import the entire project from client to server

**svn import *user_directory_path* file:///*repository_path* -m *'Message'***

9.1: SVN - Checkout

## The Working Copy (Check-Out)

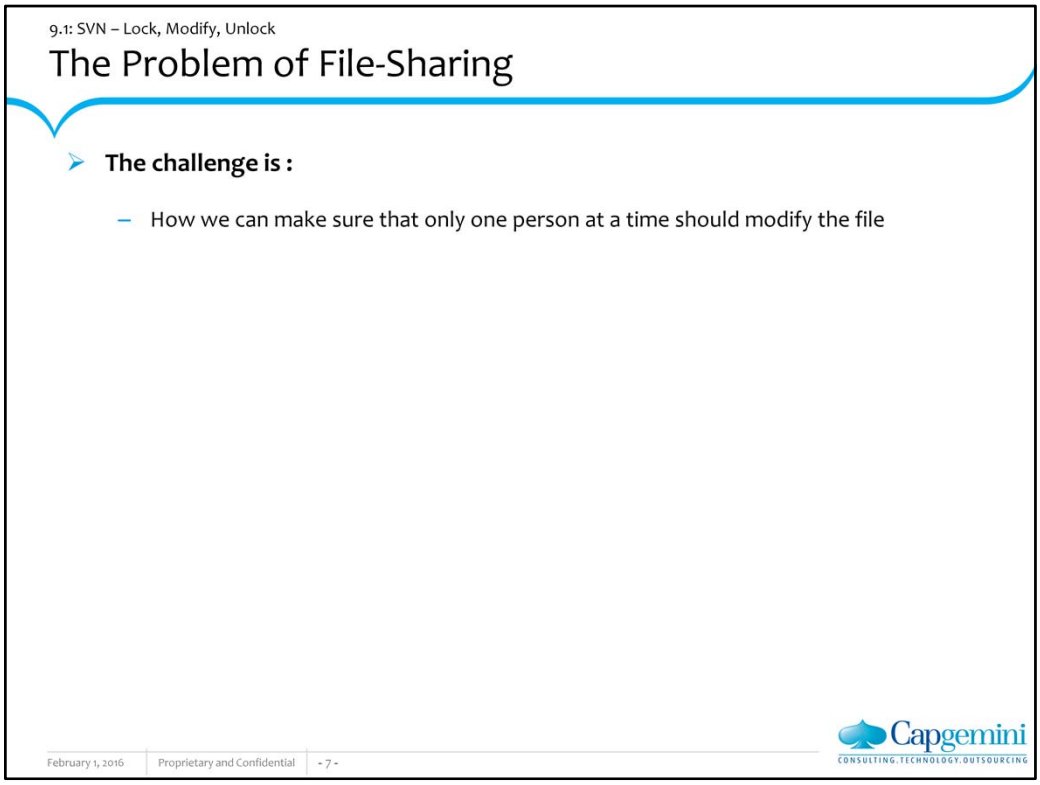
- **An ordinary directory tree on local system, containing a collection of files is called working copy.**
- **Owner's private work area ( edit , compile, change any files/code)**
  - Complete Checkout

```
svn checkout file:///repository_path
```

  - Partial Checkout

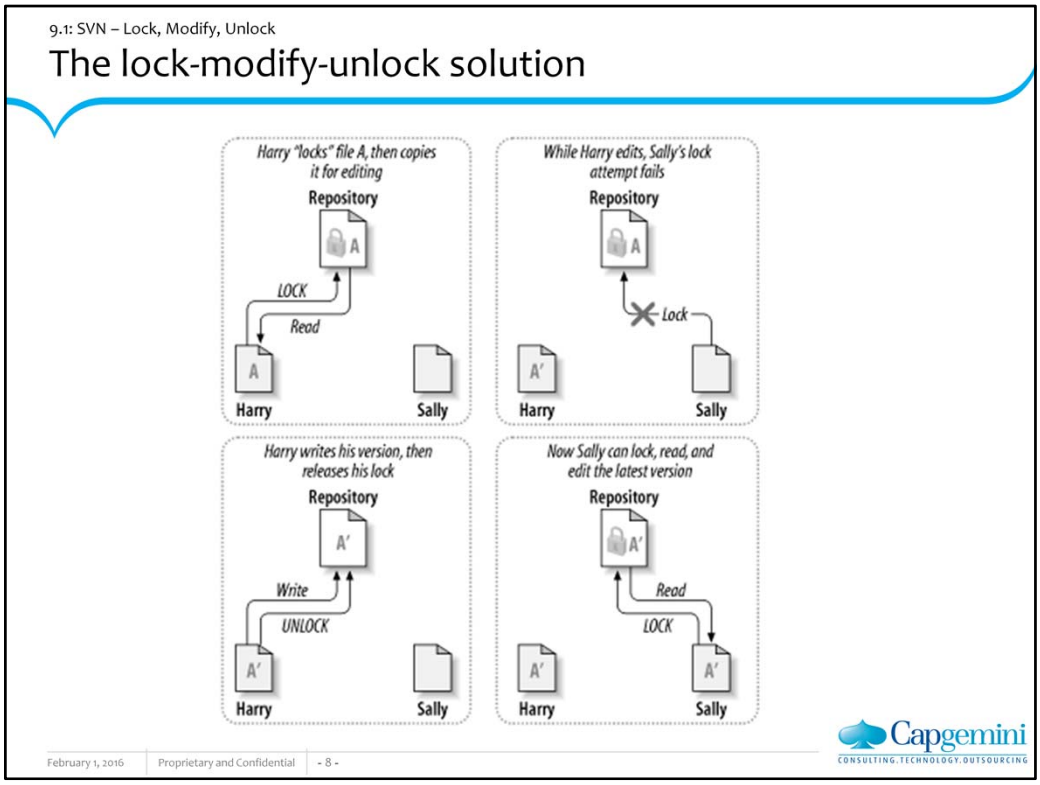
```
svn checkout file:///repository_path/Directory_Path
```

- **Changes in working copy needs explicit commit to update the changed version in repository**




All version control systems have to solve the same fundamental problem: how will the system allow users to share information, but prevent them from accidentally stepping on each other's feet? It's all too easy for users to accidentally overwrite each other's changes in the repository.

Suppose we have two co-workers, Harry and Sally. They each decide to edit the same repository file at the same time. If Harry saves his changes to the repository first, then it's possible that (a few moments later) Sally could accidentally overwrite them with her own new version of the file. While Harry's version of the file won't be lost forever (because the system remembers every change), any changes Harry made won't be present in Sally's newer version of the file, because she never saw Harry's changes to begin with. Harry's work is still effectively lost or at least missing from the latest version of the file and probably by accident. This is definitely a situation we want to avoid!




The Lock-Modify-Unlock Solution

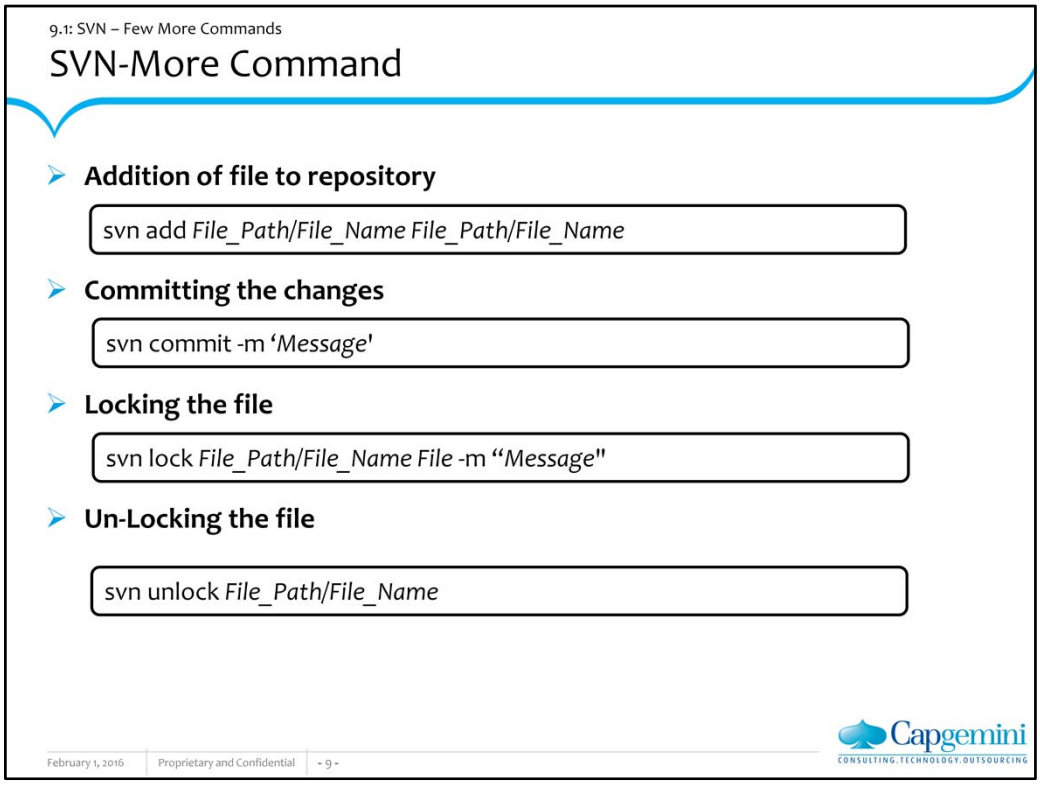
Many version control systems use a lock-modify-unlock model to address the problem of many authors clobbering each other's work. In this model, the repository allows only one person to change a file at a time. This exclusivity policy is managed using locks. Harry must lock a file before he can begin making changes to it. If Harry has locked a file, then Sally cannot also lock it, and therefore cannot make any changes to that file. All she can do is read the file, and wait for Harry to finish his changes and release his lock. After Harry unlocks the file, Sally can take her turn by locking and editing the file.

The problem with the lock-modify-unlock model is that it's a bit restrictive, and often becomes a roadblock for users:

--Locking may cause administrative problems. Sometimes Harry will lock a file and then forget about it. Meanwhile, because Sally is still waiting to edit the file, her hands are tied.

--Locking may cause unnecessary serialization. What if Harry is editing the beginning of a text file, and Sally simply wants to edit the end of the same file? These changes don't overlap at all. They could easily edit the file simultaneously.

The solution of this problem is “Copy-Modify-Merge”


9.1: SVN – Few More Commands
SVN-More Command
Addition of file to repository
svn add File_Path/File_Name File_Path/File_Name
Committing the changes
svn commit -m 'Message'
Locking the file
svn lock File_Path/File_Name File -m "Message"
Un-Locking the file
svn unlock File_Path/File_Name
February 1, 2016
Proprietary and Confidential
- 9 -
Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

## Summary

- SVN

## Review Questions

- In _____ multiple copies are not stored only changes are stored.
- List any 3 commands used in SVN?




Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

## Reference Books

- **The Unix Programming Environment**
  - Kerningham & Pike, Prentice Hall
- **Unix System V.4 Concepts and Applications**
  - Sumitabha Das, Tata McGraw-Hill
- **Advanced Unix Programmer's Guide**
  - Stephen Prata, BPB
- **Introducing Unix System V**
  - Vijay Mukhi, Tata McGrawHill