CSCI 2132
Software Development

Lecture 5:

File Permissions

Instructor: Vlado Keselj

Faculty of Computer Science

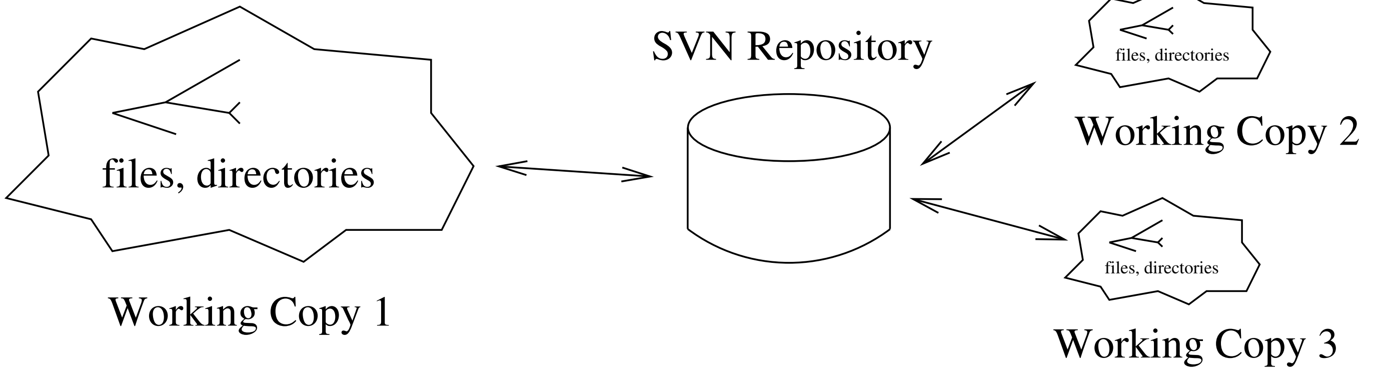
Dalhousie University

Previous Lecture

- **Files and Directories**
- Pathnames
- Commands for managing and navigating directory structure
- Commands: cat, logout, exit, ls, dirname, basename, pwd, cd, mkdir, rmdir, mv, rm, tree
- File manipulation commands
- File permissions:
  - users, groups
  - checking permissions

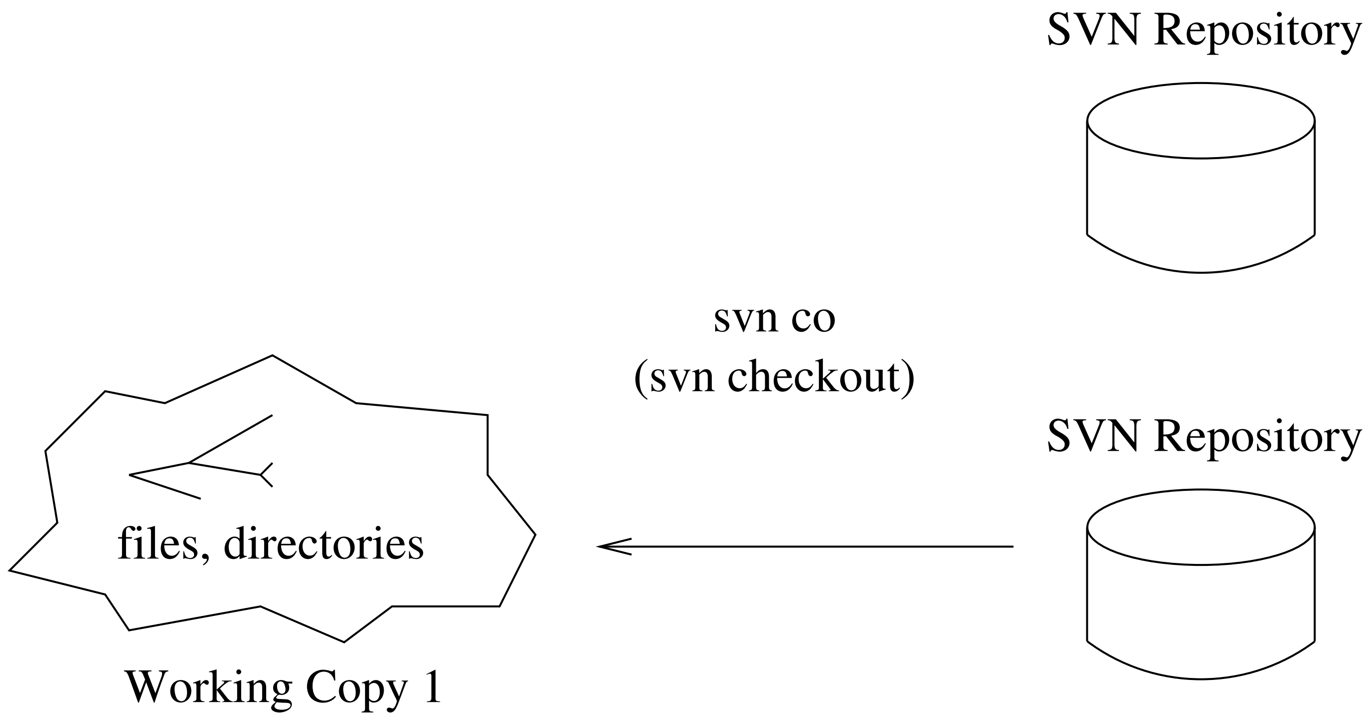
A Note About SVN

- SVN (Subversion) — Software Versioning and Revision Control System
- A simplified view:
  - Backups — creating backups in a repository
  - Historical — "time machine", labeled versions
  - Collaborative — different users can contribute and merge changes



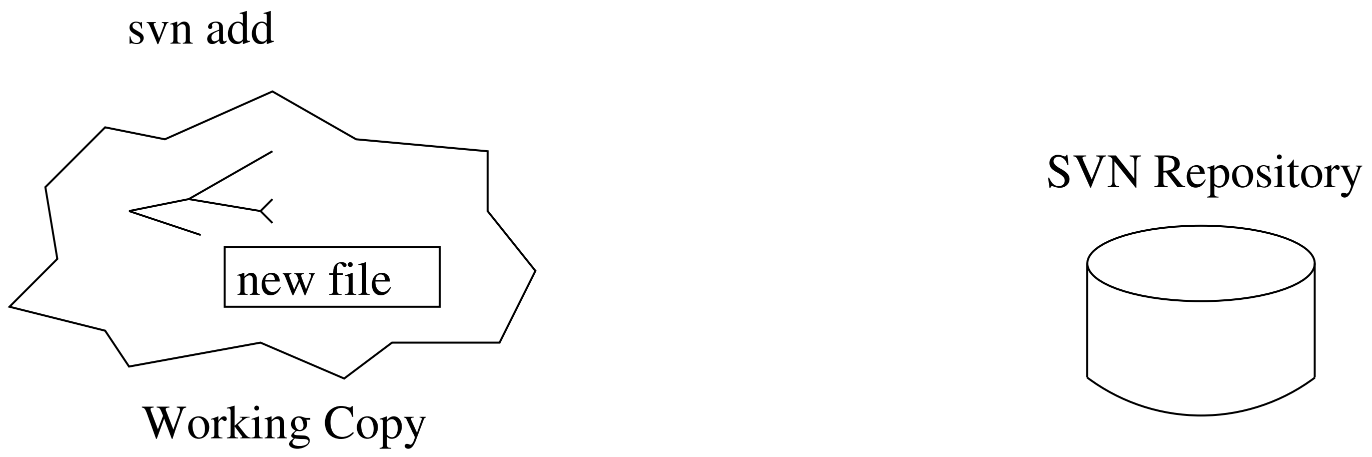
SVN Checkout (‘svn co’ or ‘svn checkout’)

- SVN checkout command is used to create an initial working copy



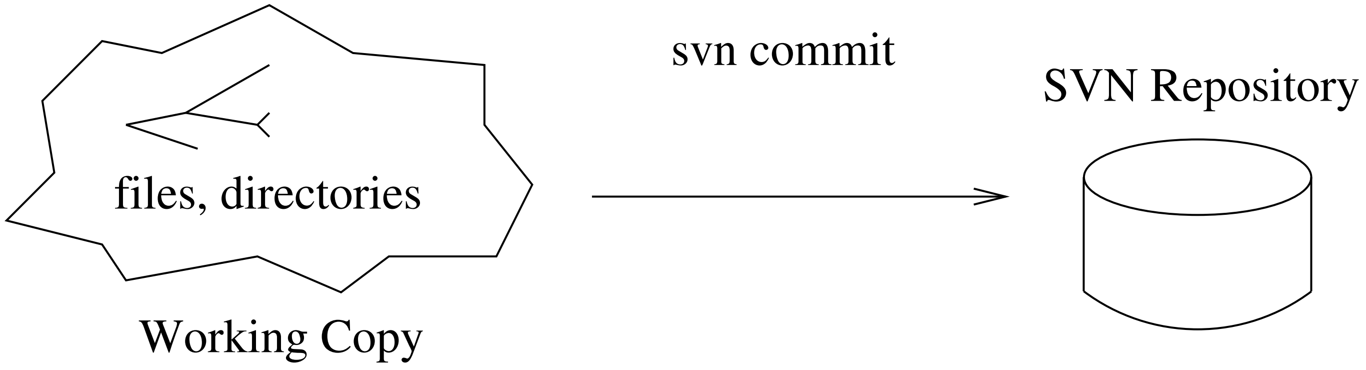
‘svn add’ command

- We can create new files in the working copy, but they are ignored by SVN
- With ‘svn add’ we add files and directories to an SVN internal list, i.e., we “mark” them not to be ignored
- The SVN repository does not know that we added files yet



‘svn commit’ command

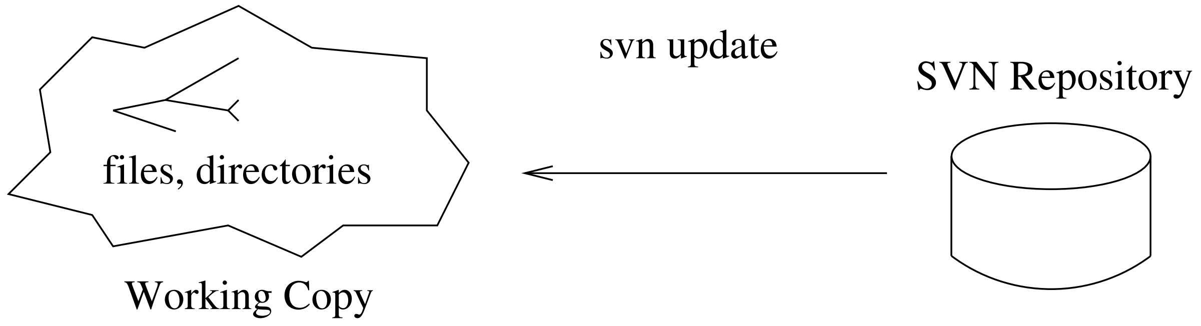
- ‘svn commit’ will save changes to the repository



- Changes are saved to the SVN repository
- Local working copy stays (you can delete it if you want, SVN repository does not need to know)
- Remember that you must provide a log message:
  ‘svn commit -mmessage’

‘svn update’ command

- It is possible that someone, or yourself, made new changes in the repository and your working copy has old versions of the files
- ‘svn update’ will update your local copy according to the changes in the repository



- It is a good idea to run ‘svn update’ if you did not modify the working copy in a long time

‘svn rm’ and ‘svn mv’

- If we remove or rename an SVN-marked file using ‘rm’ or ‘mv’, SVN will complain about it and will not remove or rename the file in the respository
- Use ‘svn rm’ to remove a file and remove it form the SVN internal list of marked files
- Use ‘svn mv’ to rename or move a file
- Changes will take affect at the next commit

SVN Troubleshooting

- Do not interrupt an SVN operation (unless it takes very long time)
- Helpful commands: ‘svn info’, ‘svn status -v’, ‘svn log -v’
- A working copy can be recognized by the hidden `.svn` directory
- A way to resolve a problem is to move or remove working copy, and make a new checked out working copy
- If you allow SVN to save your password, you can remove the record with:

```
rm ~/.subversion/auth/svn.simple/*
```

SVN and Git

- There are many Version Control Systems
- Git and SVN are probably the most popular
- Both are open-source, with a lot of similarities and some differences
- ‘svn co’ is similar to ‘git clone’
- ‘svn add’ is similar to ‘git add’
- ‘svn commit’ is similar to ‘git commit’ + ‘git push’
- ‘svn update’ is simlar to ‘git pull’
- We will cover git in more details later

A Short Note about ‘wc’

- You used ‘`wc`’ command in the lab
- `wc` stands for “word count”
- It prints the number of characters, words, and lines
- Options ‘`-c`’, ‘`-w`’, and ‘`-l`’ can be used to print only one of those numbers
- Example: `wc -c file1`
- Concepts: command, arguments, options or flags

A Short Note about Pipelines

- You were asked in the Lab to create a pipeline
- The concept of the pipeline, which belongs to the **pipe-filter software architectural pattern**
- Use pipe symbol '`|`' to connect commands to create pipeline in the Unix command-line interface
- If filename can be specified as the input file, use it only with the first command

Back to Permissions…

- We will now continue with the topic of file permissions

Octal Representation of Permissions

- Permissions can be represented with 9 bits:

$$\overbrace{\texttt{rwx}}^{\texttt{user}}\ \overbrace{\texttt{rwx}}^{\texttt{group}}\ \overbrace{\texttt{rwx}}^{\texttt{other}}$$

- For practical reasons octal system is used
- For example, what permissions are represented by octal number 750?

Checking Permissions

- Command: `ls -l`
- Note: a few more useful ls options: -a -t -r
- Example:

```
$ echo test > tmpfile.txt
$ ls -l tmpfile.txt

-rw-r--r-- 1 vlado csfac 5 Sep 13 11:21 file.txt
```

Changing Permissions

- Command: `chmod` *mode files*
- chmod — changing file mode bits
- Some examples:
  - chmod 664 file.txt
  - chmod og-r file.txt
  - chmod u+x,og+r file.txt
  - chmod u=rw,og= file.txt
  - chmod a+r file.txt
  - chmod -R u+r+w+X dir1
- Note: `a` is used for ‘all’

Changing Owner and Group of a File

- Examples:
  - `chown newuser file.txt`
  - `chown -R newuser files dirs`
  - `chgrp newgroup file.txt`
  - `chgrp -R newgroup files dirs`
- `-R` is used for directory recursive change

Effective UserID and GroupID

- How does the system decide access permission for a process?
- Each process has an effective UserID and GroupID, as well as real UserID and GroupID
- Example: our shell has our UserID and a GroupID
- How are processes assigned effective userids and groupids?

Changing Effective GroupID and UserID

- `newgrp newgroup`
  - changes into newgroup (logs into new group)
- `su newuser`
  - changes effective user
  - needs to be superuser (root user)
- Additional permission bits: setuid, setgid, and sticky bit bits

Reading

- Reading: UNIX book, Ch1 and Ch2 to page 51, so far
- The book contains tutorials on vi and emacs

Redirection and Pipes

- The three standard channels: standard input, standard output, standard error output
- Modifying channels: redirection and pipes

Output Redirection

- Remember what we learned about: stdin, stdout, stderr
- Redirecting the standard output of a program into a file: `command > filename`
- Creates a file (filename) if it does not exist
- Example: `ls lab1 > listing`
- Important: '`>`' redirection deletes previous file contents
- To append a file with new content use '`>>`'
- Example: `ls lab1 >> listing`
- Creates a file ('listing') if it does not exist, as well

Input Redirection

- Redirects the standard input from a file into a process
- Useful in testing
- Syntax: `command < filename`
- Example: `sort < names.txt`
  - sorts names in a file `names.txt` and prints out
- Example 2:

  `sort < names.txt > names-sorted.txt`
- Example 3: `mail csusername < HelloWorld.java`
- Example 4: `mail full@email < HelloWorld.java`

Error Redirection

- Standard error output is not redirected by `>`
- Syntax (bash specific):

```
command 2> filename
```

- Cannot be a space between '`2`' and '`>`'
- Example: `rm x 2> error`
- If file `x` does not exist, it will produce an error message
- `>>` can be used to append output:

```
command 2>> filename
```

More About Redirection

- File descriptors of stdin, stdout, and stderr are 0, 1, and 2, respectively
- That is where 2 comes from in error redirection
- Similarly we can use 0 and 1 in input and output redirection:

```
command 0< filename
command 1> filename
```

- These are equivalent to previous redirections