Introduction to Subversion

CSC 207 Lecture1

Imagine this…

-

Imagine this…

- 

Imagine this…

- 

"Hey, Jane, could you send me a copy of those Java files in USB that you made last Tuesday?"

Imagine this…

- 

"Hey, Jane, could you send me a copy of those Java files in USB that you made last Tuesday?"

Imagine this…

- 

"Hey, Jane, could you send me a copy of those Java files in USB that you made last Tuesday?"

"Bob, this function that you EMAILED me doesn't work anymore. Did you change something?"

Imagine this…

- 

"Hey, Jane, could you send me a copy of those Java files in USB that you made last Tuesday?"

"Bob, this function that you EMAILED me doesn't work anymore. Did you change something?"

Imagine this…

- 

"Hey, Jane, could you send me a copy of those Java files in USB that you made last Tuesday?"

"Bob, this function that you EMAILED me doesn't work anymore. Did you change something?"

"Ok, we've all been working hard for the last week. Now let's integrate everyone's work together. Damn… project does't compile anymore, are we missing any files?"

Here are all the bad points from previous slide.

Here are all the bad points from previous slide.

- Does this sound like a disorganized project?

Here are all the bad points from previous slide.

- Does this sound like a disorganized project?
- Developers email their source files to each other? **(This is bad)**

Here are all the bad points from previous slide.

- Does this sound like a disorganized project?
- Developers email their source files to each other? **(This is bad)**
- Someone is responsible for merging/combining different source files together to make it all compile and work. **(This is painful process)**

Here are all the bad points from previous slide.

- Does this sound like a disorganized project?
- Developers email their source files to each other? **(This is bad)**
- Someone is responsible for merging/combining different source files together to make it all compile and work. **(This is painful process)**
- There is no way to keep track of **what file** got changed at **what time** and by **what developer**.

What is version control?

What is version control?

- **Tracks files:**

What is version control?

- **Tracks files:**

1) over a period of time

What is version control?

- **Tracks files:**

1) over a period of time

2) by what developers.

What is version control?

- **Tracks files:**

1) over a period of time

2) by what developers.

You now know WHO wrote what part of the code and WHAT your files looked like an hour ago, a week ago, or a year ago.

What is version control?

- **Tracks files:**

1) over a period of time

2) by what developers.

You now know WHO wrote what part of the code and WHAT your files looked like an hour ago, a week ago, or a year ago.

What is version control?

- **Tracks files:**

1) over a period of time

2) by what developers.

You now know WHO wrote what part of the code and WHAT your files looked like an hour ago, a week ago, or a year ago.

- **Merge Contributions:**

What is version control?

- **Tracks files:**

1) over a period of time

2) by what developers.

You now know WHO wrote what part of the code and WHAT your files looked like an hour ago, a week ago, or a year ago.

- **Merge Contributions:**

of the many developers into a single whole.

How does version control track files?

How does version control track files?

- Simple answer: It does this by creating a revision.

How does version control track files?

- Simple answer: It does this by creating a revision.

- What is revision?

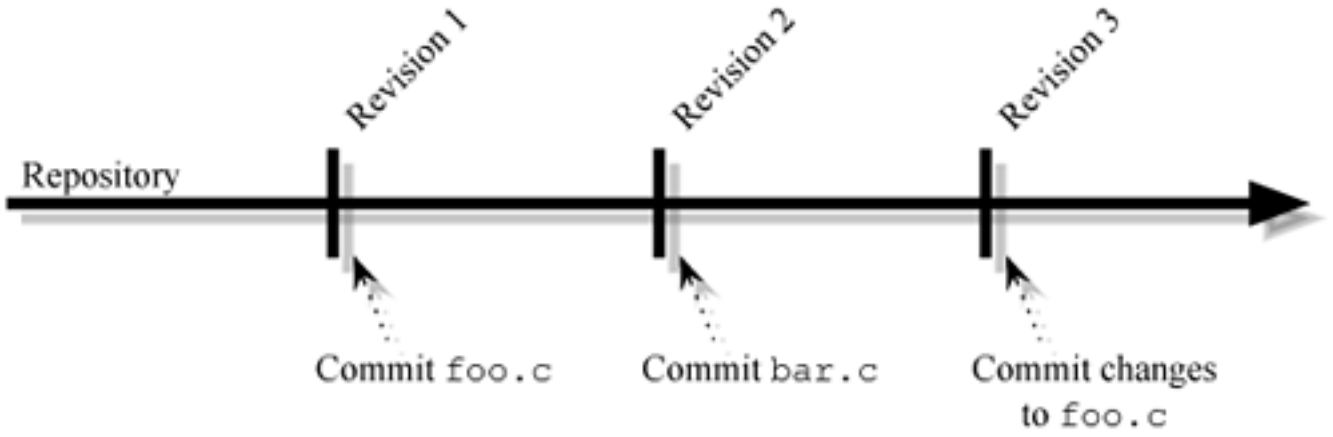
How does version control track files?

- Simple answer: It does this by creating a revision.

- What is revision?

Keep track of the changing states of files over time and merge contributions of multiple developers.

A repository (central location) with revision numbers.

- 

We will later understand that in SVN there are two components.

1) Client
2) Server

The server component of SVN is also called repository or central location.

Other benefits of version control

Other benefits of version control

- Data Integrity:

Other benefits of version control

- Data Integrity:

Nothing is ever deleted or lost, if the revision has been created.

Other benefits of version control

- Data Integrity:

Nothing is ever deleted or lost, if the revision has been created.

Other benefits of version control

- Data Integrity:

Nothing is ever deleted or lost, if the revision has been created.

Other benefits of version control

- Data Integrity:

Nothing is ever deleted or lost, if the revision has been created.

- Productivity:

Other benefits of version control

- Data Integrity:

Nothing is ever deleted or lost, if the revision has been created.

- Productivity:

Frees developers from manual integration of work.

Other benefits of version control

- Data Integrity:

Nothing is ever deleted or lost, if the revision has been created.

- Productivity:

Frees developers from manual integration of work.

If a certain revision is unstable (i.e. tests are failing) you can easily revert/go back to earlier revisions (when your code was stable).

Other benefits of version control

Other benefits of version control

- Accountability:

Other benefits of version control

- Accountability:

Allows you to answer the following questions:

Other benefits of version control

- Accountability:

Allows you to answer the following questions:

Who added each bit of code to project?

Other benefits of version control

- Accountability:

Allows you to answer the following questions:

Who added each bit of code to project?

When did this developer add the change?

Other benefits of version control

- Accountability:

Allows you to answer the following questions:

Who added each bit of code to project?

When did this developer add the change?

Who has made modifications since then?

Other benefits of version control

- Accountability:

Allows you to answer the following questions:

Who added each bit of code to project?

When did this developer add the change?

Who has made modifications since then?

- Record Keeping

Other benefits of version control

- Accountability:

Allows you to answer the following questions:

Who added each bit of code to project?

When did this developer add the change?

Who has made modifications since then?

- Record Keeping
- Rapid Development

Some examples of version control

-

Some examples of version control

Some examples of version control

Some examples of version control

Some examples of version control

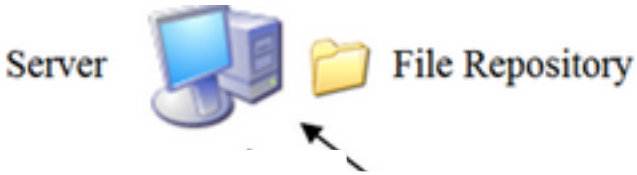
What is client-server architecture?

- Some examples from real life?

Client-Server Architecture of Subversion (SVN)

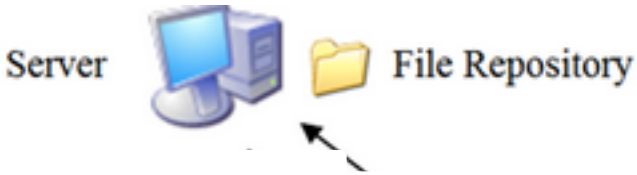
- 

Client-Server Architecture of Subversion (SVN)

- 

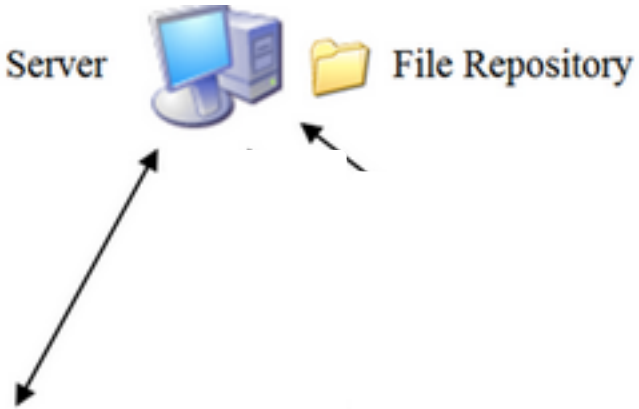
Client-Server Architecture of Subversion (SVN)

- **For this semester, SVN server is on CSLinux @ UTM**



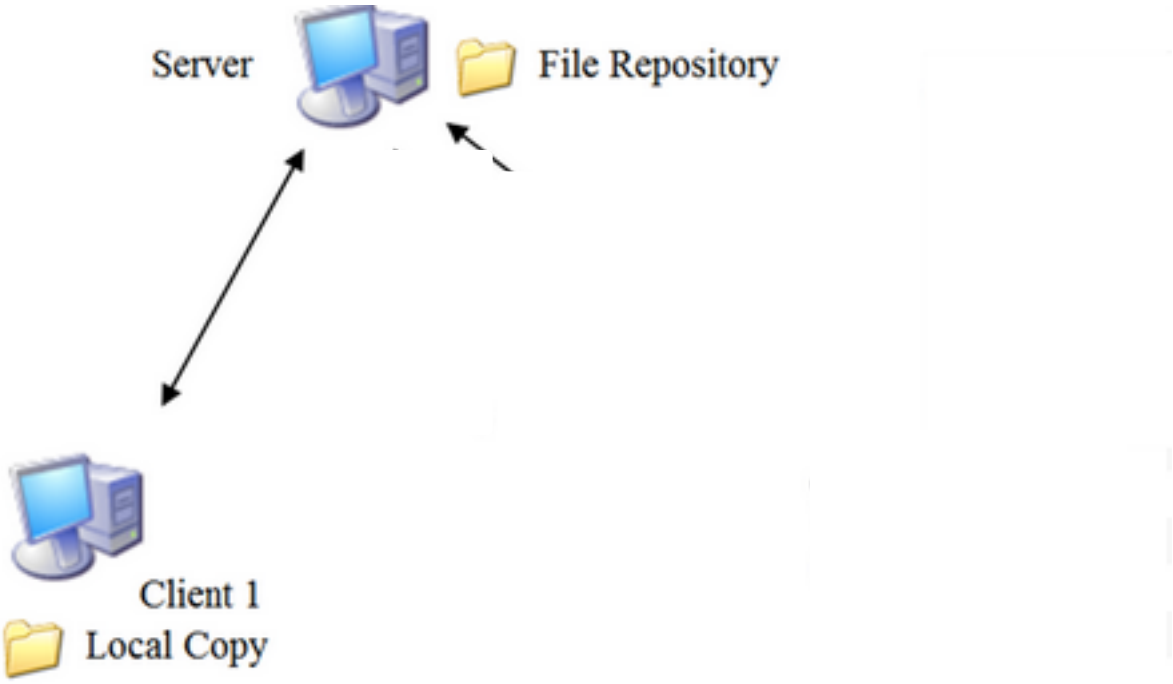
Client-Server Architecture of Subversion (SVN)

- **For this semester, SVN server is on CSLinux @ UTM**



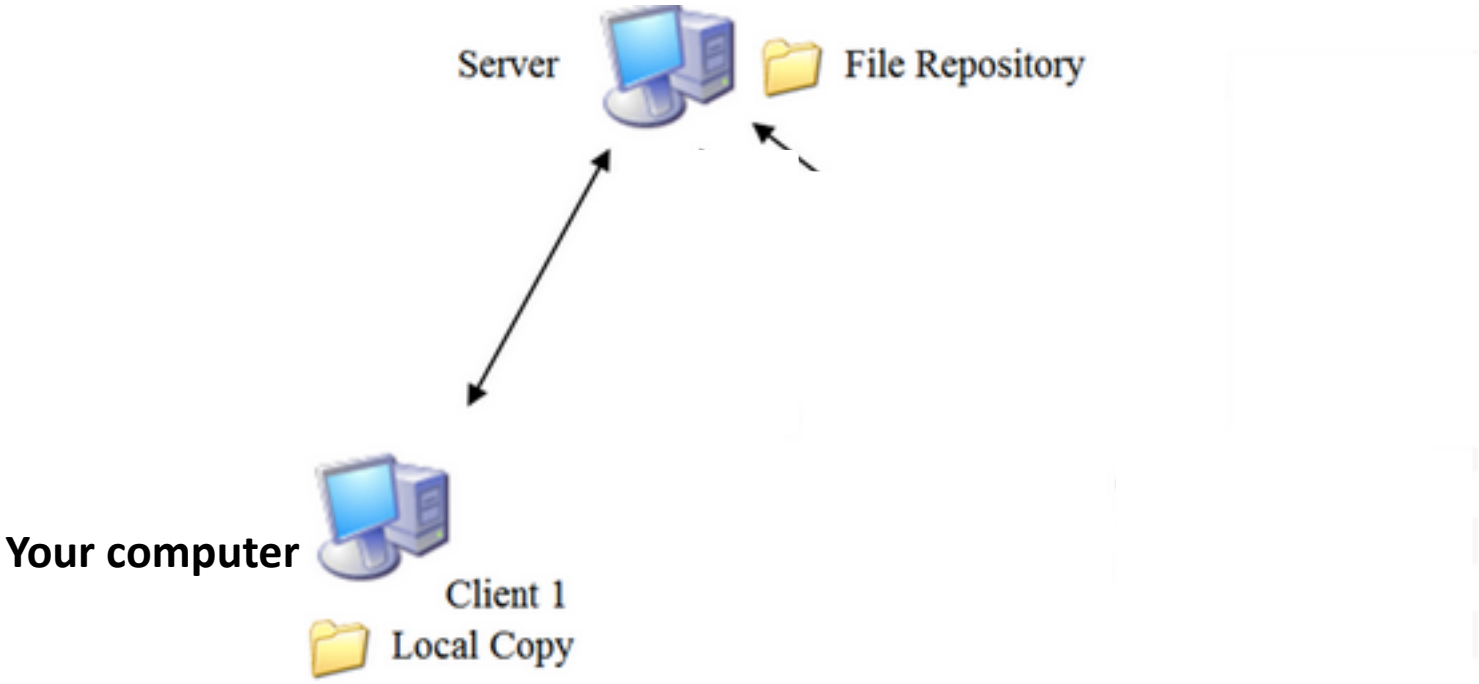
Client-Server Architecture of Subversion (SVN)

- **For this semester, SVN server is on CSLinux @ UTM**



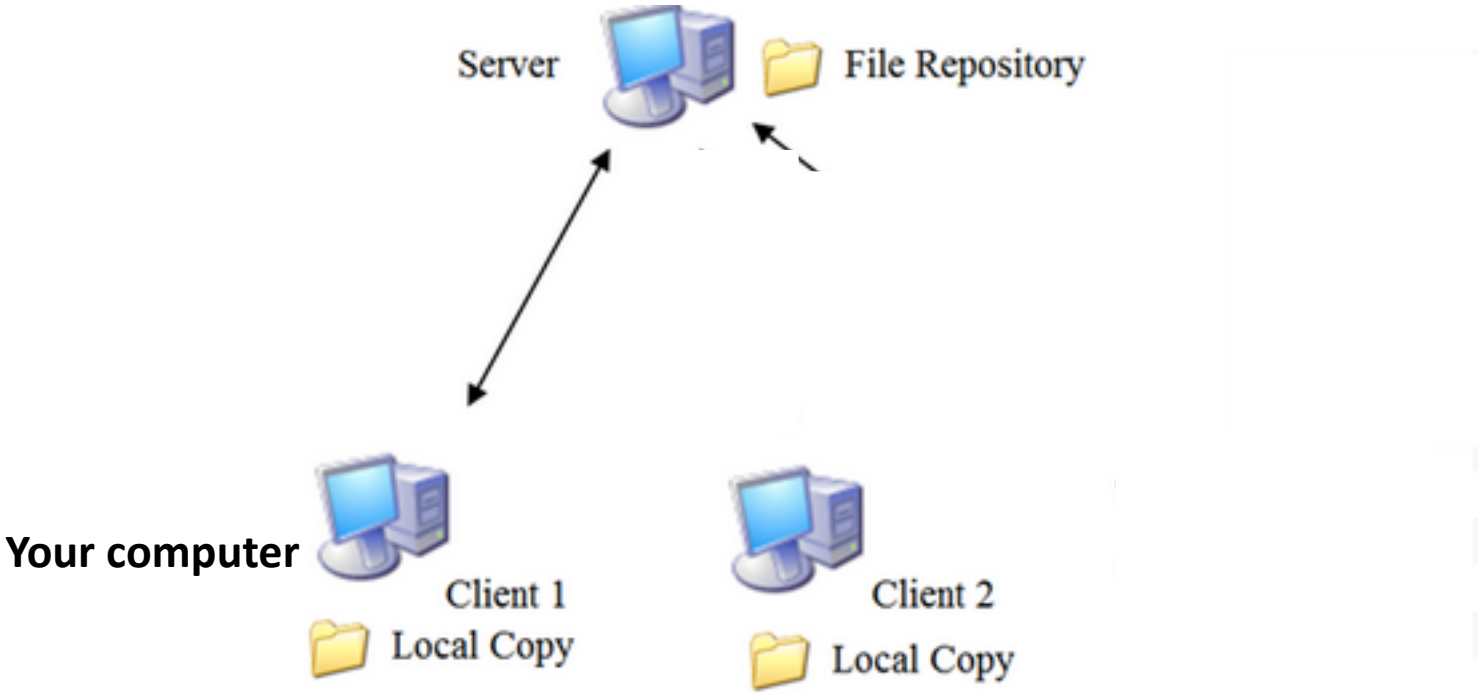
Client-Server Architecture of Subversion (SVN)

- **For this semester, SVN server is on CSLinux @ UTM**



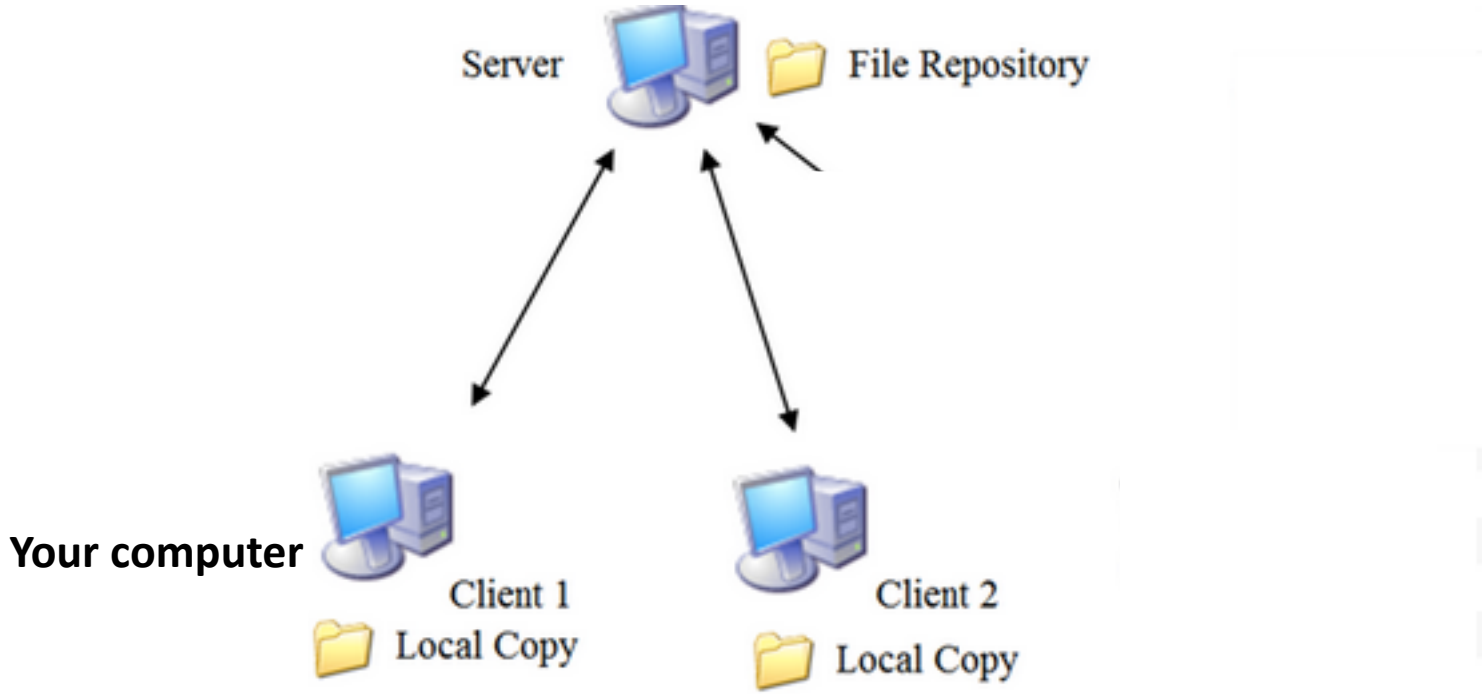
Client-Server Architecture of Subversion (SVN)

- **For this semester, SVN server is on CSLinux @ UTM**



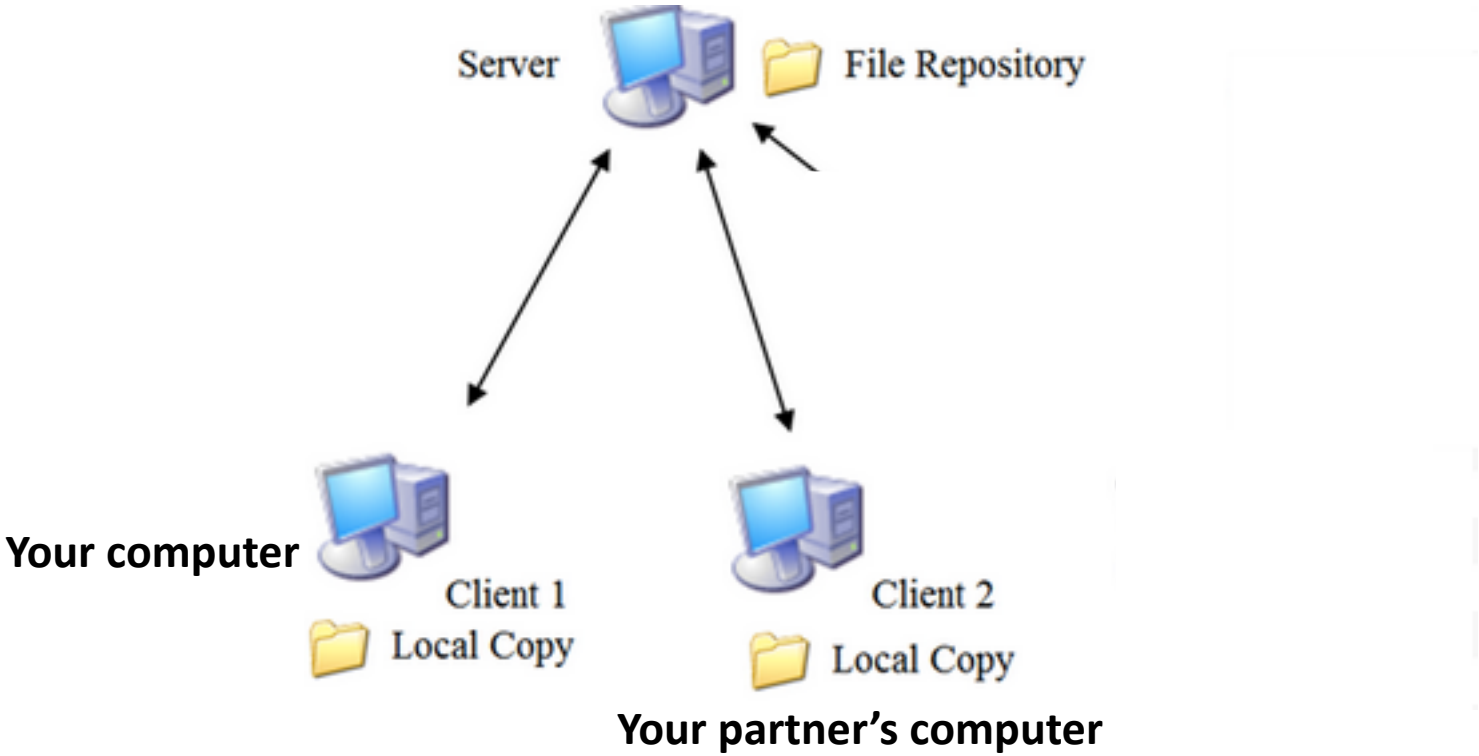
Client-Server Architecture of Subversion (SVN)

- **For this semester, SVN server is on CSLinux @ UTM**



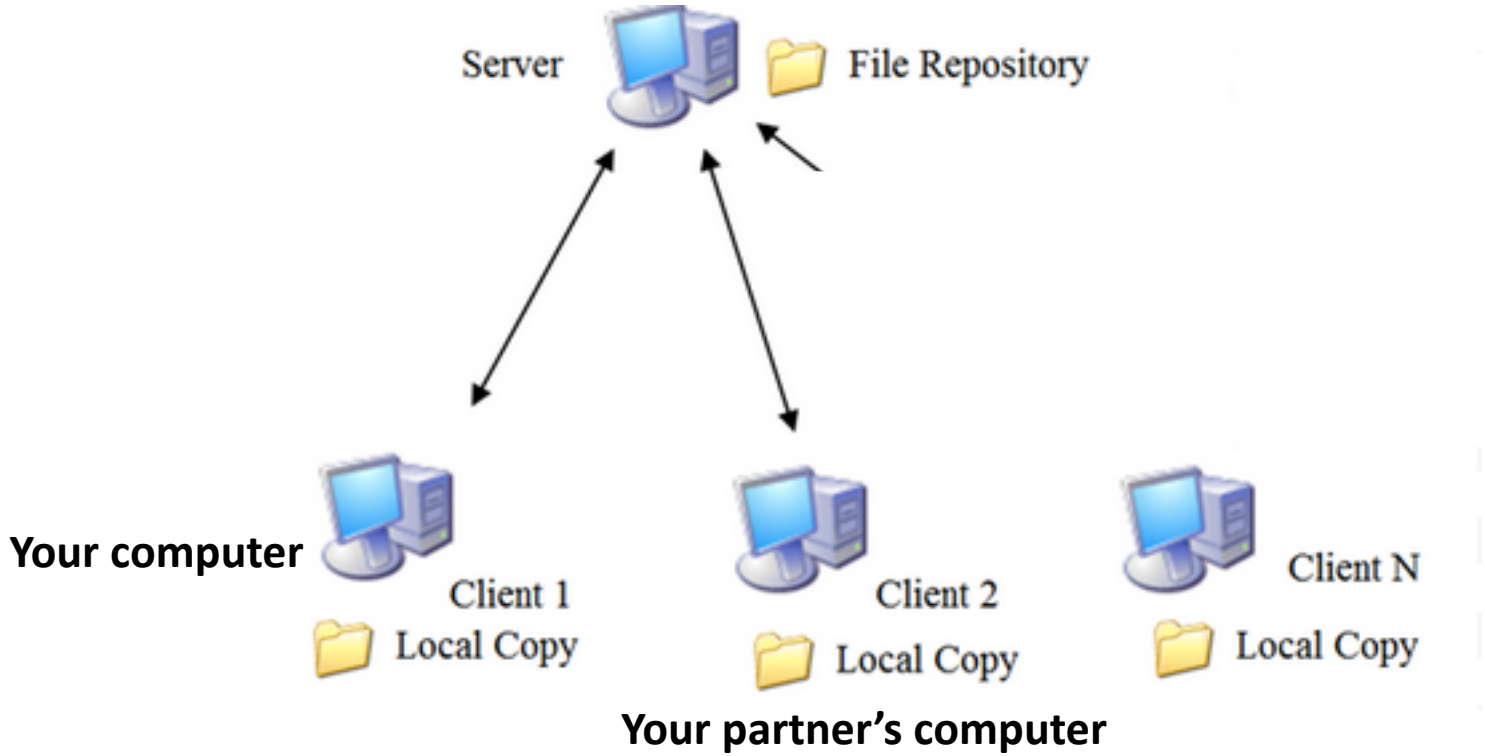
Client-Server Architecture of Subversion (SVN)

- **For this semester, SVN server is on CSLinux @ UTM**



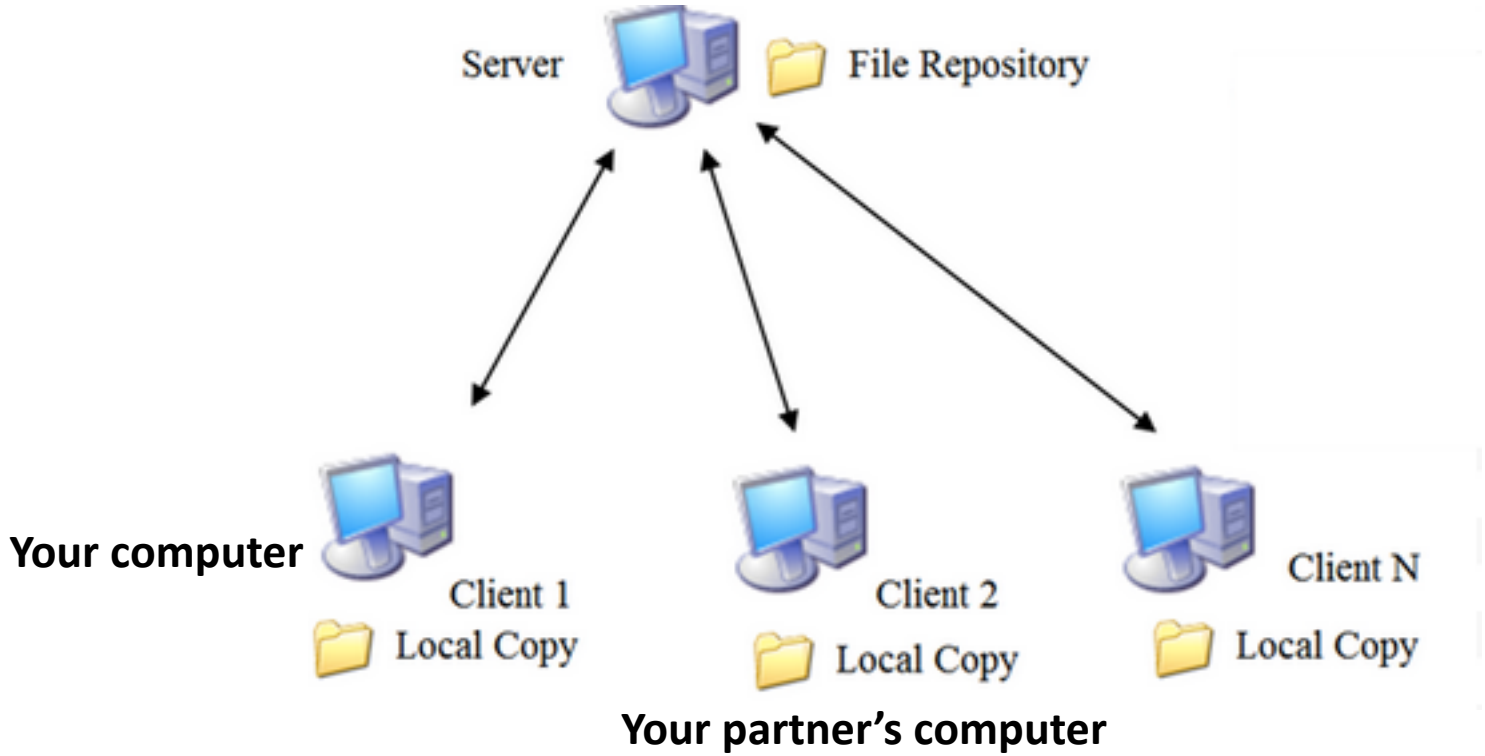
Client-Server Architecture of Subversion (SVN)

- **For this semester, SVN server is on CSLinux @ UTM**



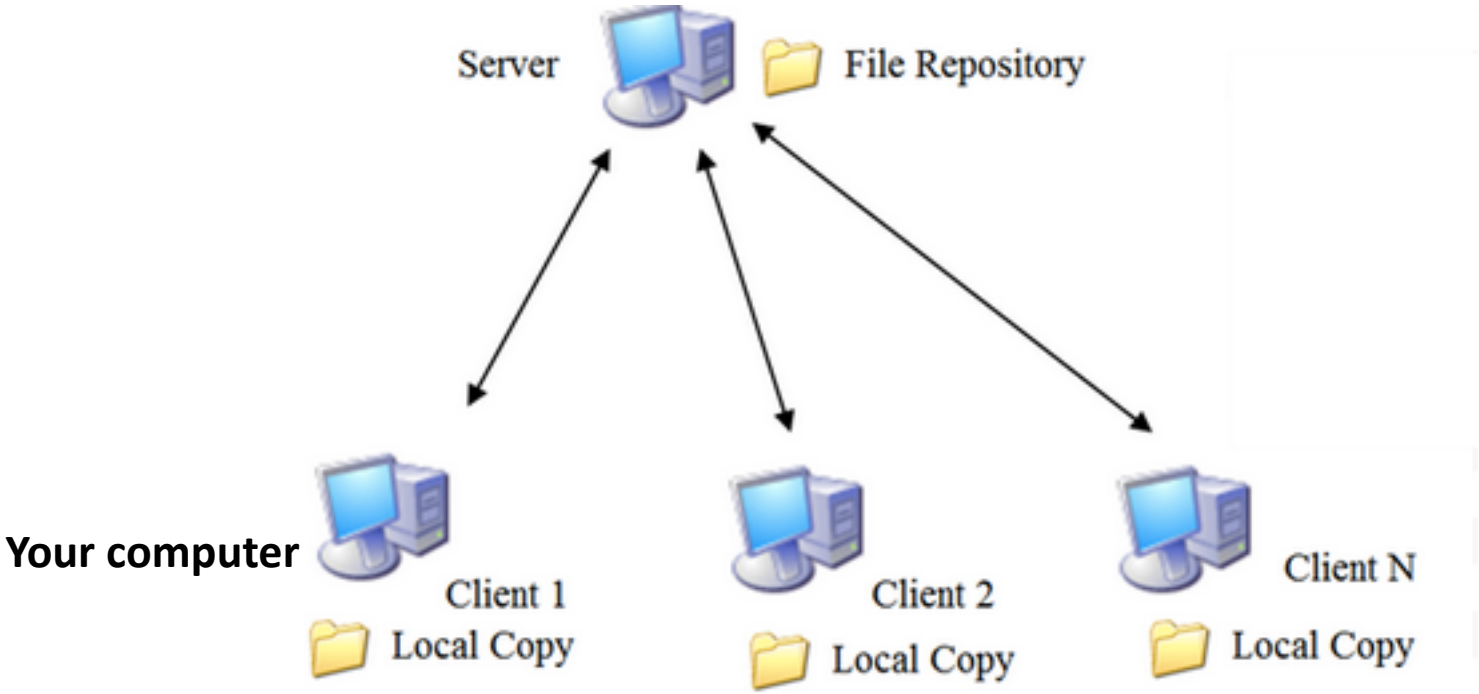
Client-Server Architecture of Subversion (SVN)

- **For this semester, SVN server is on CSLinux @ UTM**



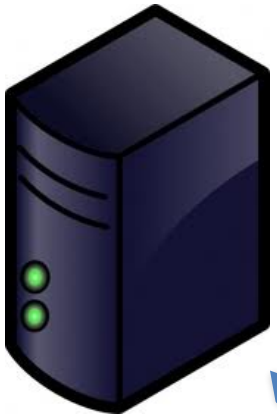
Client-Server Architecture of Subversion (SVN)

- **For this semester, SVN server is on CSLinux @ UTM**



Demo Time.

1) We have two new people that are joining a software company, lets call them Male and Female



SVN Server
--Central storage of your files

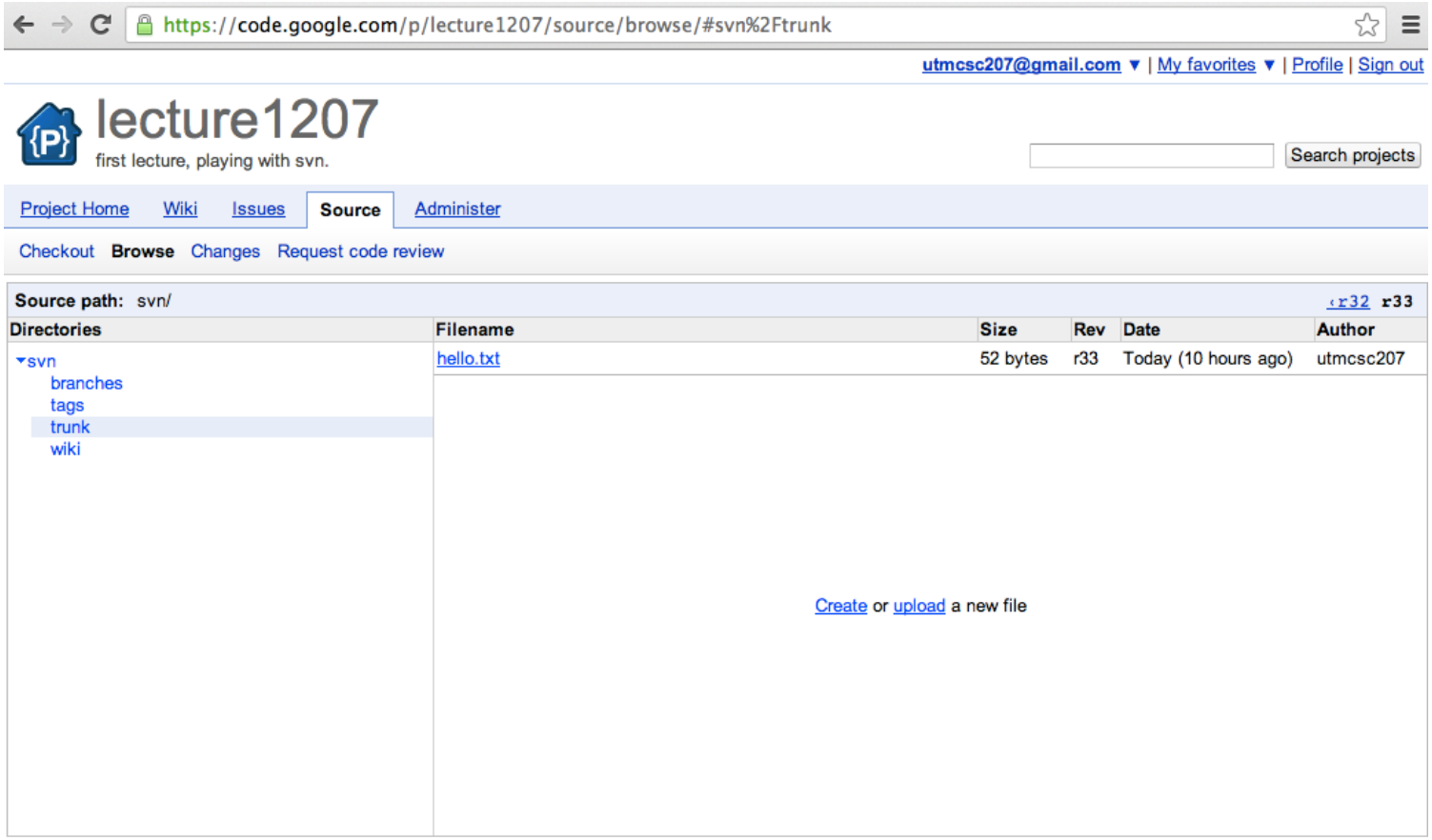
https://lecture1207.googlecode.com/svn/trunk/

--SVN Client
--Creates a working copy

--SVN Client
--Creates a working copy

What is currently on the svn server?

- Lets say the company has the following files on the server:



- Developers do not work directly with the files on the server.

- Developers do not work directly with the files on the server.

- Developers do not work directly with the files on the server.
- Developers instead use SVN client to connect to the server, and checkout the files onto their computer.

- Developers do not work directly with the files on the server.

- Developers instead use SVN client to connect to the server, and checkout the files onto their computer.
  - This process of getting files onto your computer creates a **working directory**.

- Developers do not work directly with the files on the server.

- Developers instead use SVN client to connect to the server, and checkout the files onto their computer.
  - This process of getting files onto your computer creates a **working directory**.
  - Working directory is a directory on your computer that contains the contents of the server.

- Developers do not work directly with the files on the server.

- Developers instead use SVN client to connect to the server, and checkout the files onto their computer.
  - This process of getting files onto your computer creates a **working directory**.
  - Working directory is a directory on your computer that contains the contents of the server.

- Developers do not work directly with the files on the server.

- Developers instead use SVN client to connect to the server, and checkout the files onto their computer.
  - This process of getting files onto your computer creates a **working directory**.
  - Working directory is a directory on your computer that contains the contents of the server.

- Developers now make changes or add new files to the working directory which at some point can be *pushed* to the server.

- Developers do not work directly with the files on the server.

- Developers instead use SVN client to connect to the server, and checkout the files onto their computer.
  - This process of getting files onto your computer creates a **working directory**.
  - Working directory is a directory on your computer that contains the contents of the server.

- Developers now make changes or add new files to the working directory which at some point can be *pushed* to the server.
- Working directory can easily be updated (*pull*) when new content is available on the server.

What is SVN client?

What is SVN client?

- SVN client is a piece of software that runs on the developer's computer AND that allows to connect to the server (svn server) so that files can be *pulled* onto the local computer or *pushed* onto the server.

What is SVN client?

- SVN client is a piece of software that runs on the developer's computer AND that allows to connect to the server (svn server) so that files can be *pulled* onto the local computer or *pushed* onto the server.

What is SVN client?

- SVN client is a piece of software that runs on the developer's computer AND that allows to connect to the server (svn server) so that files can be *pulled* onto the local computer or *pushed* onto the server.

- Linux and OS-X computers have a command line version of svn client on their computers. There are multiple GUI svn clients available (svn plugin for Eclipse and svn plugin for NetBeans) as well which can be downloaded.

The very first time…

- Our friend ‘Male’, wishes to connect to the company’s svn server, and pull all the contents from it and create a working copy on his computer.

```
Abbass-MacBook-Air:sandboxA abbas$ pwd
/Users/abbas/Desktop/sandboxA
Abbass-MacBook-Air:sandboxA abbas$
```

What is svn checkout?

- This is what our friend 'MALE' types on the command line.

(This needs to be done only once when ever you like to create a working copy)

svn checkout https://lecture1207.googlecode.com/svn/trunk/ lecture1207 --username xyzM@gmail.com

svn -> this is the svn client.
checkout-> this is the argument to the svn client for creating a working copy.
https://lecture1207.googlecode.com/svn/trunk/ -> this argument is the SVN server location.
--lecture1207->this argument specifies what folder on your current working directory will be created. Ofcourse this could be any name that you wish.
--username -> this argument specifies the username
xyzM@gmail.com -> the actual username

What is svn checkout?

- This is what our friend 'MALE' types on the command line.

(This needs to be done only once when ever you like to create a working copy)

svn checkout https://lecture1207.googlecode.com/svn/trunk/ lecture1207 --username xyzM@gmail.com

svn -> this is the svn client.
checkout-> this is the argument to the svn client for creating a working copy.
https://lecture1207.googlecode.com/svn/trunk/ -> this argument is the SVN server location.
--lecture1207->this argument specifies what folder on your current working directory will be created. Ofcourse this could be any name that you wish.
--username -> this argument specifies the username
xyzM@gmail.com -> the actual username

What is svn checkout?

- This is what our friend 'MALE' types on the command line.

(This needs to be done only once when ever you like to create a working copy)

svn checkout https://lecture1207.googlecode.com/svn/trunk/ lecture1207 --username xyzM@gmail.com

svn -> this is the svn client.
checkout-> this is the argument to the svn client for creating a working copy.
https://lecture1207.googlecode.com/svn/trunk/ -> this argument is the SVN server location.
--lecture1207->this argument specifies what folder on your current working directory will be created. Ofcourse this could be any name that you wish.
--username -> this argument specifies the username
xyzM@gmail.com -> the actual username

What is svn checkout?

- This is what our friend 'MALE' types on the command line.

(This needs to be done only once when ever you like to create a working copy)

svn checkout https://lecture1207.googlecode.com/svn/trunk/ lecture1207 --username xyzM@gmail.com

svn -> this is the svn client.
checkout-> this is the argument to the svn client for creating a working copy.
https://lecture1207.googlecode.com/svn/trunk/ -> this argument is the SVN server location.
--lecture1207->this argument specifies what folder on your current working directory will be created. Ofcourse this could be any name that you wish.
--username -> this argument specifies the username
xyzM@gmail.com -> the actual username

What is svn checkout?

- This is what our friend 'MALE' types on the command line.

(This needs to be done only once when ever you like to create a working copy)

svn checkout https://lecture1207.googlecode.com/svn/trunk/ lecture1207 --username xyzM@gmail.com

svn -> this is the svn client.
checkout-> this is the argument to the svn client for creating a working copy.
https://lecture1207.googlecode.com/svn/trunk/ -> this argument is the SVN server location.
--lecture1207->this argument specifies what folder on your current working directory will be created. Ofcourse this could be any name that you wish.
--username -> this argument specifies the username
xyzM@gmail.com -> the actual username

What is svn checkout?

- This is what our friend 'MALE' types on the command line.

(This needs to be done only once when ever you like to create a working copy)

svn checkout https://lecture1207.googlecode.com/svn/trunk/ lecture1207 --username xyzM@gmail.com

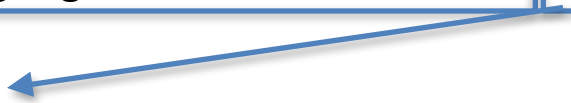
svn -> this is the svn client.
checkout-> this is the argument to the svn client for creating a working copy.
https://lecture1207.googlecode.com/svn/trunk/ -> this argument is the SVN server location.
--lecture1207->this argument specifies what folder on your current working directory will be created. Ofcourse this could be any name that you wish.
--username -> this argument specifies the username
xyzM@gmail.com -> the actual username

What is svn checkout?

- This is what our friend 'MALE' types on the command line.

(This needs to be done only once when ever you like to create a working copy)

svn checkout https://lecture1207.googlecode.com/svn/trunk/ lecture1207 --username xyzM@gmail.com



svn -> this is the svn client.
checkout-> this is the argument to the svn client for creating a working copy.
https://lecture1207.googlecode.com/svn/trunk/ -> this argument is the SVN server location.
--lecture1207->this argument specifies what folder on your current working directory will be created. Ofcourse this could be any name that you wish.
--username -> this argument specifies the username
xyzM@gmail.com -> the actual username

What is svn checkout?

- This is what our friend 'MALE' types on the command line.

(This needs to be done only once when ever you like to create a working copy)

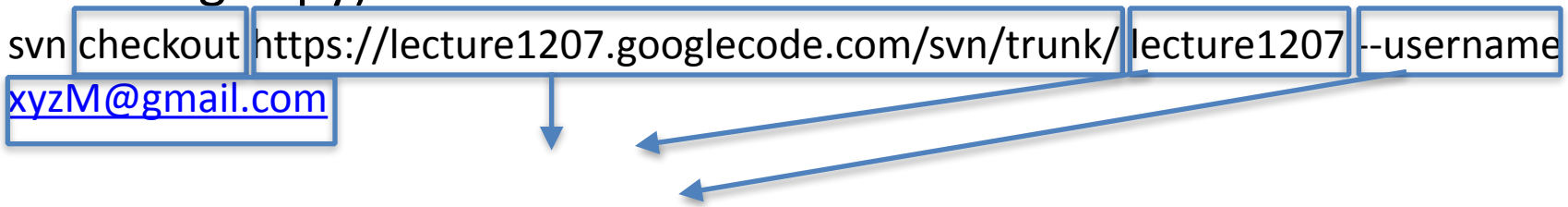
svn checkout https://lecture1207.googlecode.com/svn/trunk/ lecture1207 --username xyzM@gmail.com

svn -> this is the svn client.
checkout-> this is the argument to the svn client for creating a working copy.
https://lecture1207.googlecode.com/svn/trunk/ -> this argument is the SVN server location.
--lecture1207->this argument specifies what folder on your current working directory will be created. Ofcourse this could be any name that you wish.
--username -> this argument specifies the username
xyzM@gmail.com -> the actual username

What is svn checkout?

- This is what our friend 'MALE' types on the command line.

(This needs to be done only once when ever you like to create a working copy)

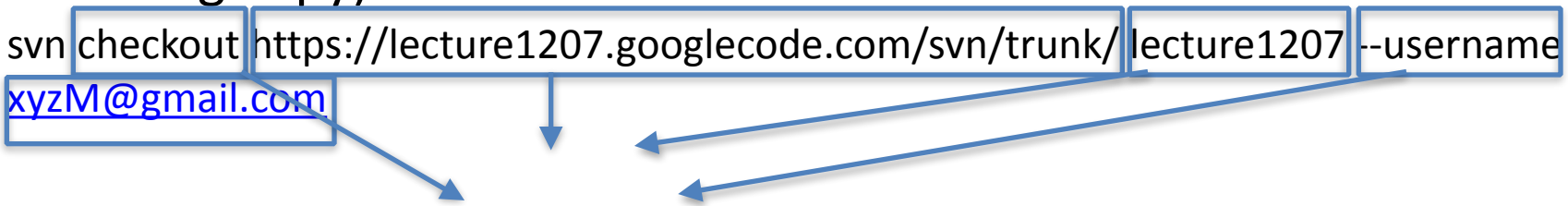


svn -> this is the svn client.
checkout-> this is the argument to the svn client for creating a working copy.
https://lecture1207.googlecode.com/svn/trunk/ -> this argument is the SVN server location.
--lecture1207->this argument specifies what folder on your current working directory will be created. Ofcourse this could be any name that you wish.
--username -> this argument specifies the username
xyzM@gmail.com -> the actual username

What is svn checkout?

- This is what our friend 'MALE' types on the command line.

(This needs to be done only once when ever you like to create a working copy)

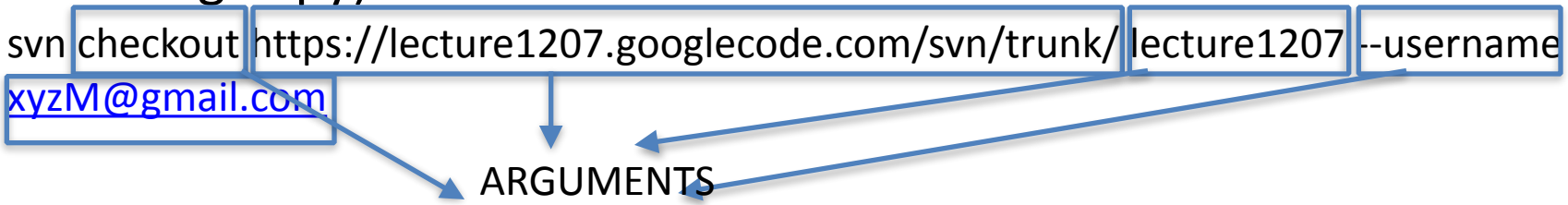


svn -> this is the svn client.
checkout-> this is the argument to the svn client for creating a working copy.
https://lecture1207.googlecode.com/svn/trunk/ -> this argument is the SVN server location.
--lecture1207->this argument specifies what folder on your current working directory will be created. Ofcourse this could be any name that you wish.
--username -> this argument specifies the username
xyzM@gmail.com -> the actual username

What is svn checkout?

- This is what our friend 'MALE' types on the command line.

(This needs to be done only once when ever you like to create a working copy)

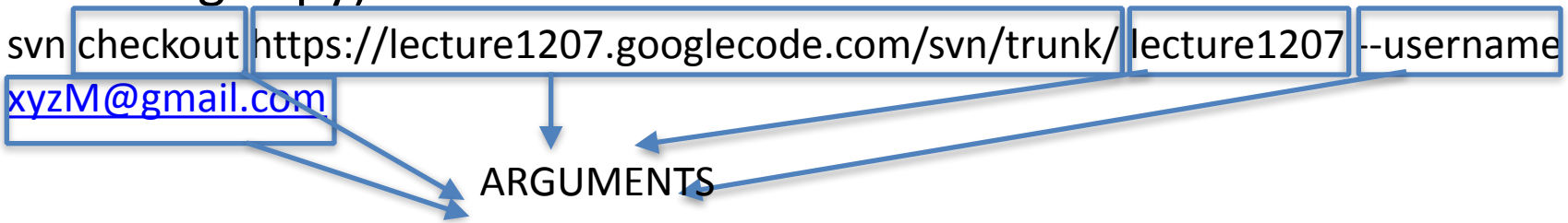


svn -> this is the svn client.
checkout-> this is the argument to the svn client for creating a working copy.
https://lecture1207.googlecode.com/svn/trunk/ -> this argument is the SVN server location.
--lecture1207->this argument specifies what folder on your current working directory will be created. Ofcourse this could be any name that you wish.
--username -> this argument specifies the username
xyzM@gmail.com -> the actual username

What is svn checkout?

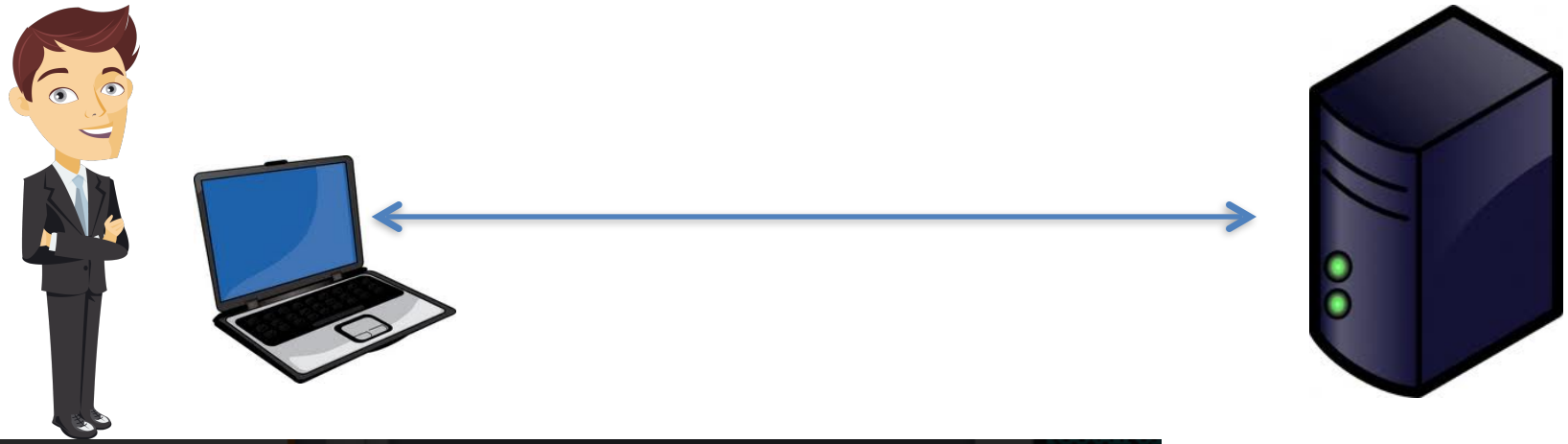
- This is what our friend 'MALE' types on the command line.

(This needs to be done only once when ever you like to create a working copy)



svn -> this is the svn client.
checkout-> this is the argument to the svn client for creating a working copy.
https://lecture1207.googlecode.com/svn/trunk/ -> this argument is the SVN server location.
--lecture1207->this argument specifies what folder on your current working directory will be created. Ofcourse this could be any name that you wish.
--username -> this argument specifies the username
xyzM@gmail.com -> the actual username

This is the result on the computer of ‘MALE’

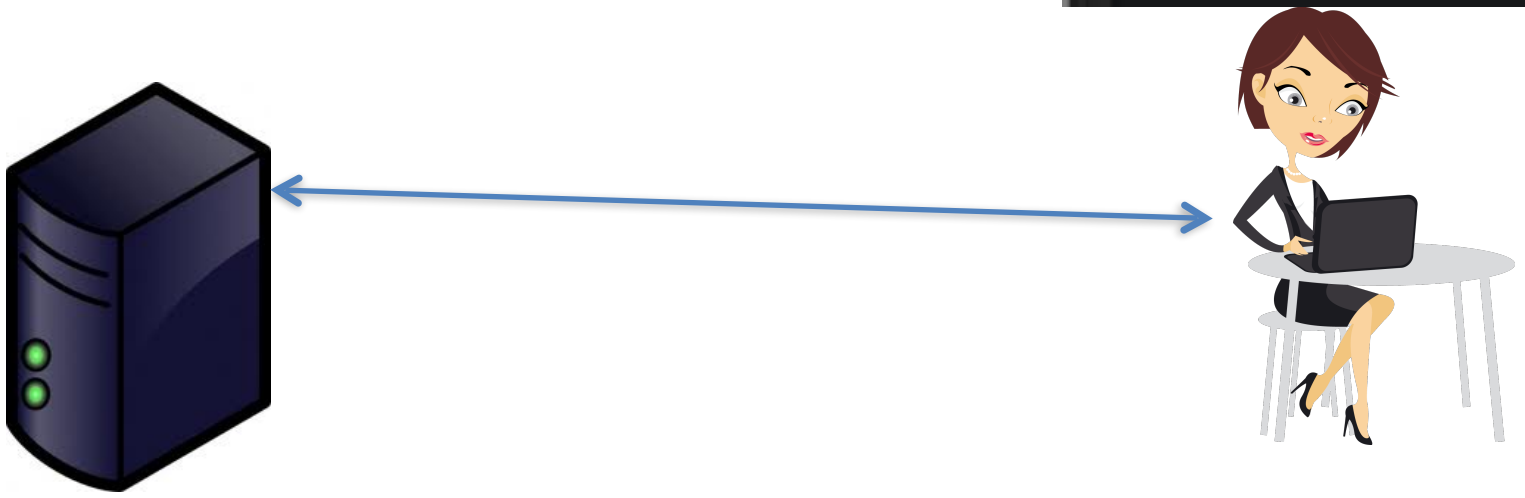
- 

```
Abbass-MacBook-Air:sandboxA abbas$ svn checkout https://lecture1207.googlecode.c
om/svn/trunk/ lecture1207 --username utmcsc207@gmail.com
A    lecture1207/hello.txt
Checked out revision 33.
Abbass-MacBook-Air:sandboxA abbas$ ls
lecture1207
Abbass-MacBook-Air:sandboxA abbas$ cd lecture1207/
Abbass-MacBook-Air:lecture1207 abbas$ ls
hello.txt
Abbass-MacBook-Air:lecture1207 abbas$ pwd
/Users/abbas/Desktop/sandboxA/lecture1207
Abbass-MacBook-Air:lecture1207 abbas$
```

The very first time…

- Our friend ‘Female’, wishes to connect to the company’s svn server, and pull all the contents from it and create a working copy on her computer.

```
Abbass-MacBook-Air:sandboxB abbas$ pwd
/Users/abbas/Desktop/sandboxB
Abbass-MacBook-Air:sandboxB abbas$
```



What is svn checkout?

- This is what our friend 'Female' types on the command line.

(This needs to be done only once when ever you like to create a working copy)

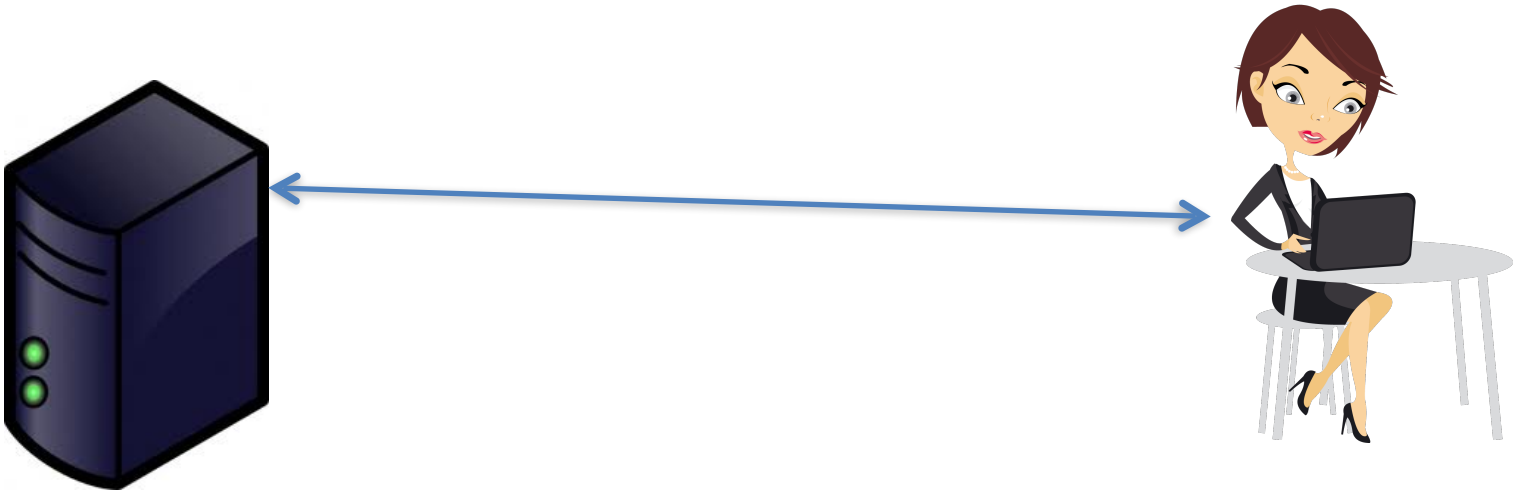
svn checkout https://lecture1207.googlecode.com/svn/trunk/ lecture1207 --username xyzF@gmail.com

svn -> this is the svn client.
checkout-> this is the argument to the svn client for creating a working copy.
https://lecture1207.googlecode.com/svn/trunk/ -> this argument is the SVN server location.
lecture1207->this argument specifies what folder on your current working directory will be created. Ofcourse this could be any name that you wish.
--username -> this argument specifies the username
xyzF@gmail.com -> the actual username

This is the result

-

```
Abbass-MacBook-Air:sandboxB abbas$ svn checkout https://lecture1207.googlecode.c
om/svn/trunk/ lecture1207 --username utmcsc207@gmail.com
A    lecture1207/hello.txt
Checked out revision 33.
Abbass-MacBook-Air:sandboxB abbas$ cd lecture1207/
Abbass-MacBook-Air:lecture1207 abbas$ ls
hello.txt
Abbass-MacBook-Air:lecture1207 abbas$ pwd
/Users/abbas/Desktop/sandboxB/lecture1207
Abbass-MacBook-Air:lecture1207 abbas$
```



Create a new file and add it to svn server

- 1) Our friend ‘MALE’ creates a local file (lets call it names.txt) on his computer INSIDE HIS WORKING DIRECTORY
- 2) MALE enters some names in the file.
- 3) The following command prepares the file for adding to the svn server BUT NOT YET PUSHED i.e.

svn add names.txt

- 4) The following command finally pushes the file to the server i.e.

svn commit –m “adding a new file!!!” names.txt

How does our friend ‘female’ get the names.txt?

- 1) Inside her working directory she types the following command:

svn update

The above command allows the svn client on the Female’s computer to pull any files that are new or modified files on the server but not present in her working directory

What is **svn status**?

- This command prints the status of working directories and files. If you have made local changes, it will show your locally modified items.

What is **svn log**?

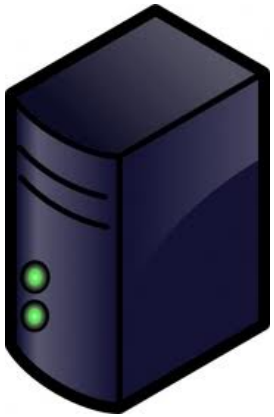
1) Shows log messages from the repository.

2) If no arguments are supplied, **svn log** shows the log messages for all files and directories inside (and including) the current working directory of your working copy

What is merging?

- Merging happens on the client and never on the server.

- Lets see an example.



Both are working on the same file i.e. ‘hello.txt’ BUT ON DIFFERENT LINES.

What are hello.txt contents on the server?

- Line1:this is a bad line
- Line2:this is a bad line

Lets imagine that *line1* and *line2* contain a bug.

Responsible for fixing line1

Responsible for fixing line2

```
Line1:this is a bad line
Line2:this is a good line
```

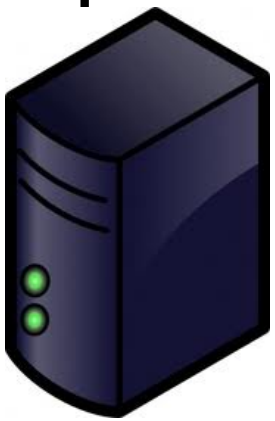
RESULT OF STEP 1

STEP 1

```
Abbass-MacBook-Air:lecture1207 abbas$ cat hello.txt
Line1:this is a bad line
Line2:this is a good line
Abbass-MacBook-Air:lecture1207 abbas$ svn commit -m "I fixed the bug!!" hello.tx
t
```

Male does not do update, but fixes line1

-



```
Line1:this is a bad line
Line2:this is a good line
```

1

```
Abbass-MacBook-Air:lecture1207 abbas$ cat hello.txt
Line1:this is a good line
Line2:this is a bad line
Abbass-MacBook-Air:lecture1207 abbas$ svn commit -m "i fixed my bug as well!!"
ello.txt
svn commit -m "i fixed my bug as wellcat hello.txt " hello.txt
Sending        hello.txt
svn: Commit failed (details follow):
svn: File or directory 'hello.txt' is out of date; try updating
svn: resource out of date; try updating
```

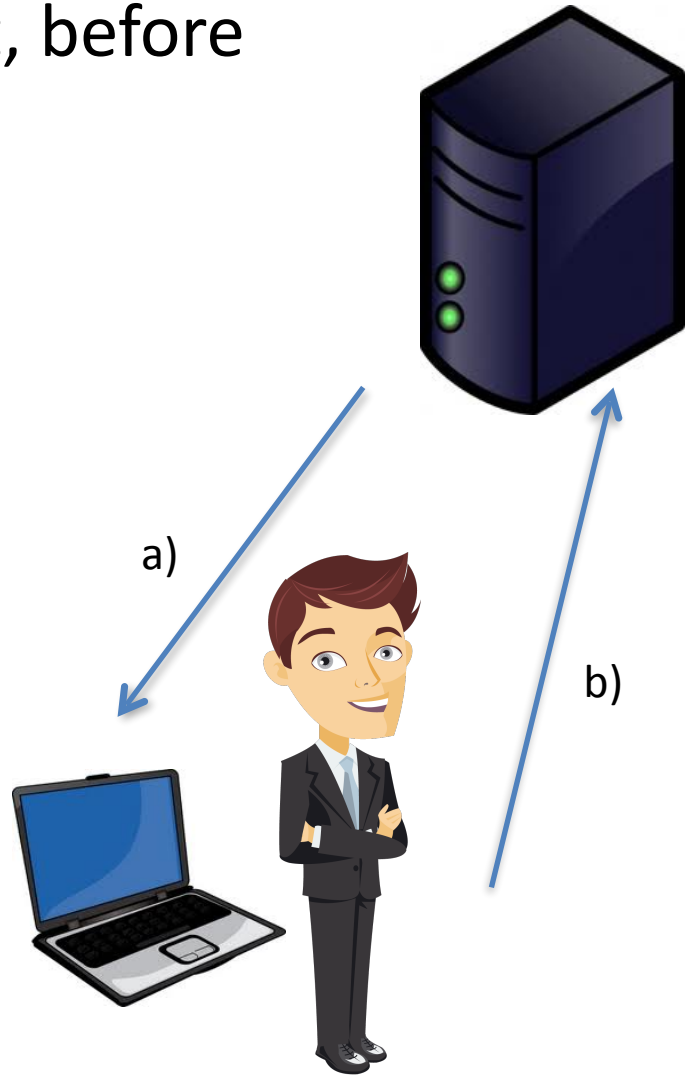
What has gone wrong here?

- Male has to do the following first, before committing.

a) svn update
Why the above command?
Did a merge happen?

AND then

b) svn commit
Why the above command?

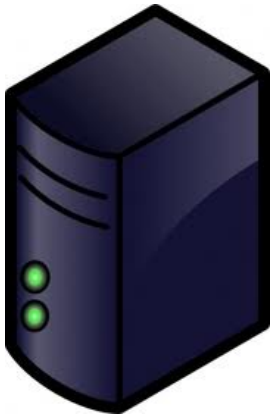


How does Female get the changes from Male?

- Female just has to type the following on the command line:

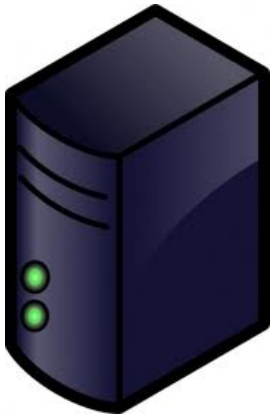
svn update

What are conflicts?



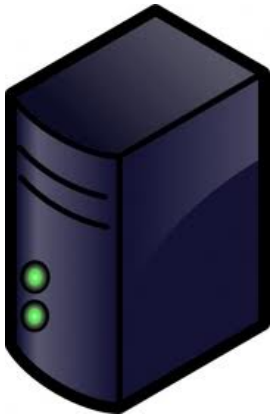
Both are working on the same file i.e. ‘hello.txt’ AND on same lines.

What are conflicts?



1) Revision x

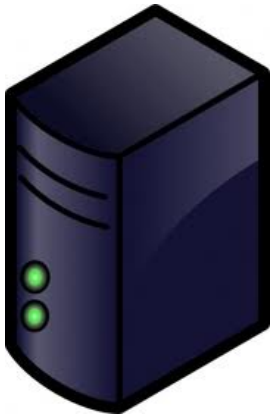
What are conflicts?



What are conflicts?

1) Revision x

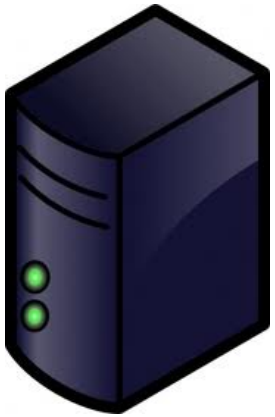
```
Line1:this is a good line
Line2:this is a good line
```



What are conflicts?

1) Revision x

```
Line1:this is a good line
Line2:this is a good line
```

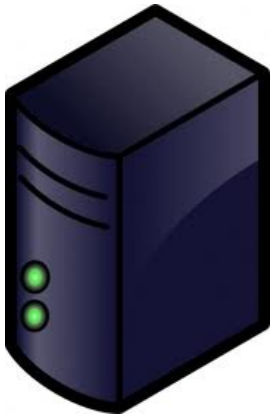


2)

What are conflicts?

1) Revision x

```
Line1:this is a good line
Line2:this is a good line
```



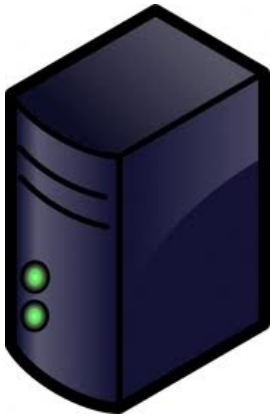
2)

```
Line1:this is a best line
Line2:this is a good line
```

What are conflicts?

1) Revision x

```
Line1:this is a good line
Line2:this is a good line
```



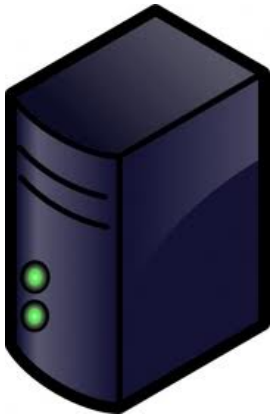
2)

```
Line1:this is a best line
Line2:this is a good line
Abbass-MacBook-Air:lecture1207 abbas$ svn commit -m "i made line1 best" hello.tx
t
Sending        hello.txt
Transmitting file data .
Committed revision 37.
```

What are conflicts?

1) Revision x

```
Line1:this is a good line
Line2:this is a good line
```



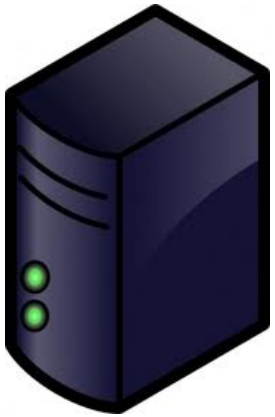
2)

```
Line1:this is a best line
Line2:this is a good line
Abbass-MacBook-Air:lecture1207 abbas$ svn commit -m "i made line1 best" hello.tx
t
Sending        hello.txt
Transmitting file data .
Committed revision 37.
```

What are conflicts?

1) Revision x

```
Line1:this is a good line
Line2:this is a good line
```



3) Revision x+1

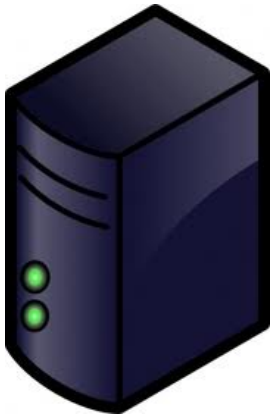
2)

```
Line1:this is a best line
Line2:this is a good line
Abbass-MacBook-Air:lecture1207 abbas$ svn commit -m "i made line1 best" hello.tx
t
Sending        hello.txt
Transmitting file data .
Committed revision 37.
```

What are conflicts?

1) Revision x

```
Line1:this is a good line
Line2:this is a good line
```



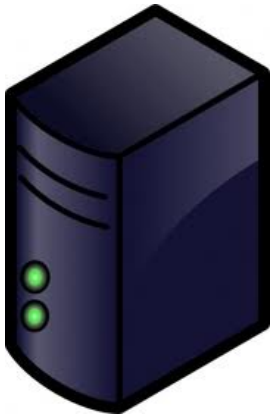
3) Revision x+1

```
Line1:this is a best line
Line2:this is a good line
```

2)

```
Line1:this is a best line
Line2:this is a good line
Abbass-MacBook-Air:lecture1207 abbas$ svn commit -m "i made line1 best" hello.tx
t
Sending        hello.txt
Transmitting file data .
Committed revision 37.
```

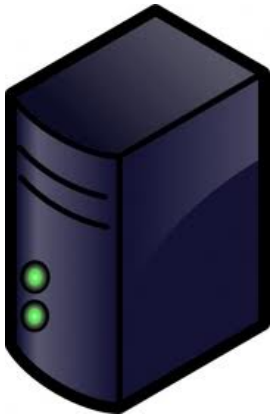
What are conflicts?



2)

1) Revision x+1

What are conflicts?

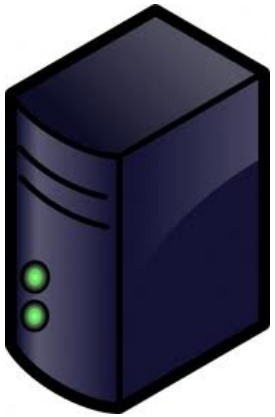


2)

What are conflicts?

1) Revision x+1

```
Line1:this is a best line
Line2:this is a good line
```

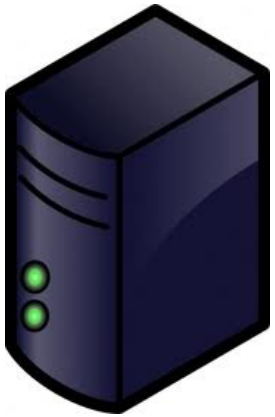


2)

What are conflicts?

1) Revision x+1

```
Line1:this is a best line
Line2:this is a good line
```



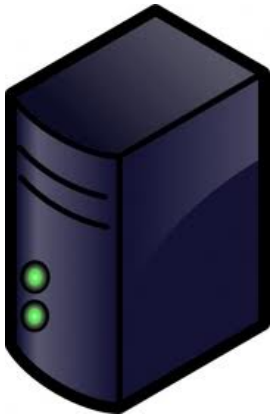
2)

```
Line1:this is a better line
Line2:this is a good line
```

What are conflicts?

1) Revision x+1

```
Line1:this is a best line
Line2:this is a good line
```

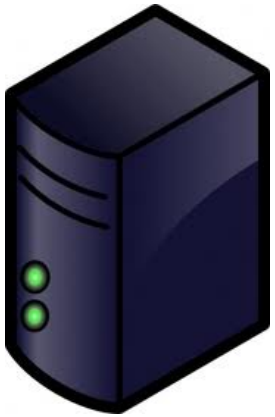


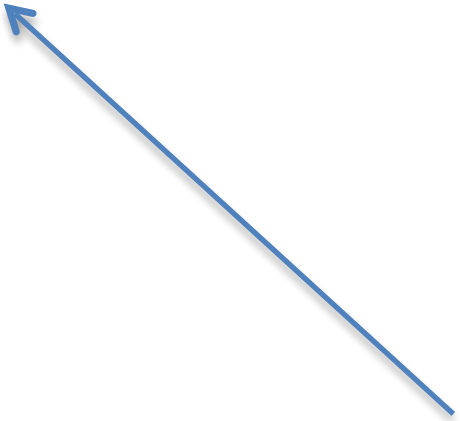
2)

```
Line1:this is a better line
Line2:this is a good line
```

```
Abbass-MacBook-Air:lecture1207 abbas$ svn commit -m "making line1 from good to b
etter"
Sending        hello.txt
svn: Commit failed (details follow):
svn: File or directory 'hello.txt' is out of date; try updating
svn: resource out of date; try updating
Abbass-MacBook-Air:lecture1207 abbas$
```

What will **svn update** do?

- 



```
Abbass-MacBook-Air:lecture1207 abbas$ svn update
Conflict discovered in 'hello.txt'.
Select: (p) postpone, (df) diff-full, (e) edit,
        (mc) mine-conflict, (tc) theirs-conflict,
        (s) show all options:
```

SVN client cannot do merge when changes overlap

- Because Male and Female were both working on the same lines of hello.txt and their changes overlap svn client on the Male cannot do automatic merge.

- Female was the first to commit into the server. Male is the last to commit to the server. The conflict happens only on the Male's computer.

How do you resolve a conflict?

- Both Male and Female need to **communicate (phone, email, or meeting in person)** and decide among them what is the best choice.

- Once they arrive at a solution, Male is responsible for resolving the conflict by either accepting the Female changes or rejecting hers and accepting his OR doing some manual merge of his and hers combination.

How does Male do this?

In order to accept female changes and discard his:

svn revert

In order to accept his changes and discard female's:

cp someFile.mine.java someFile.java
svn resolved
svn commit -m "I accepted mine changes"