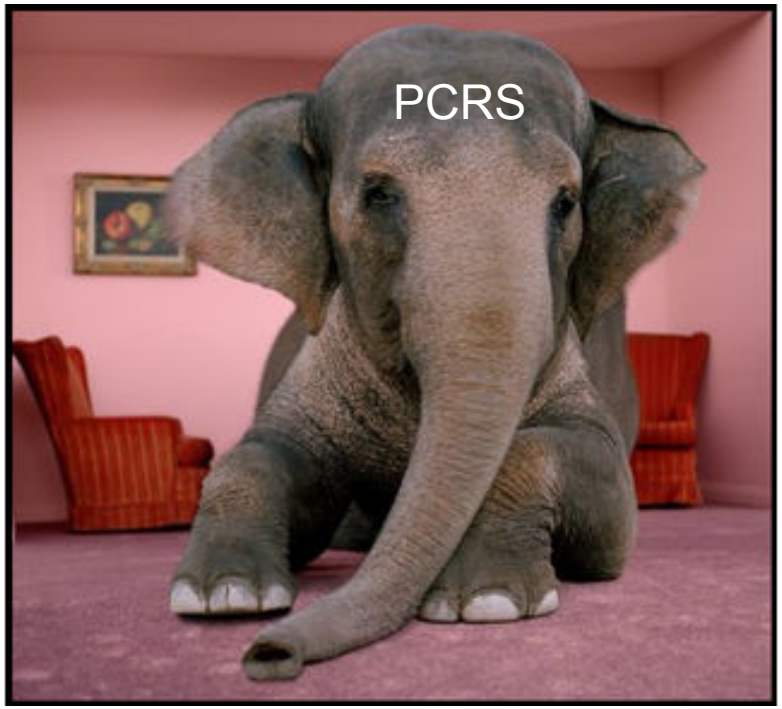
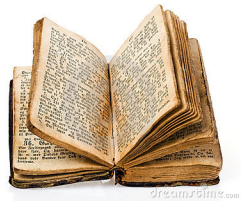
CSCB09: Software Tools and Systems Programming

Bianca Schroeder
bianca@cs.toronto.edu
IC 460

How is everyone doing?



- PCRS is not perfect. It's been developed by instructors and students for free.
- But better than the alternative: Assigning readings from the text book for class prep.



PCRS

- Some things to keep in mind:
  - Do not touch the lines of code that are already provided. Start below.
  - If that does not work try refresh.
  - For longer code segments, write your program in a file and compile/test with gcc.

Feedback

- Come and visit me in office hours.
- There is a form for anonymous feedback on the course web page.

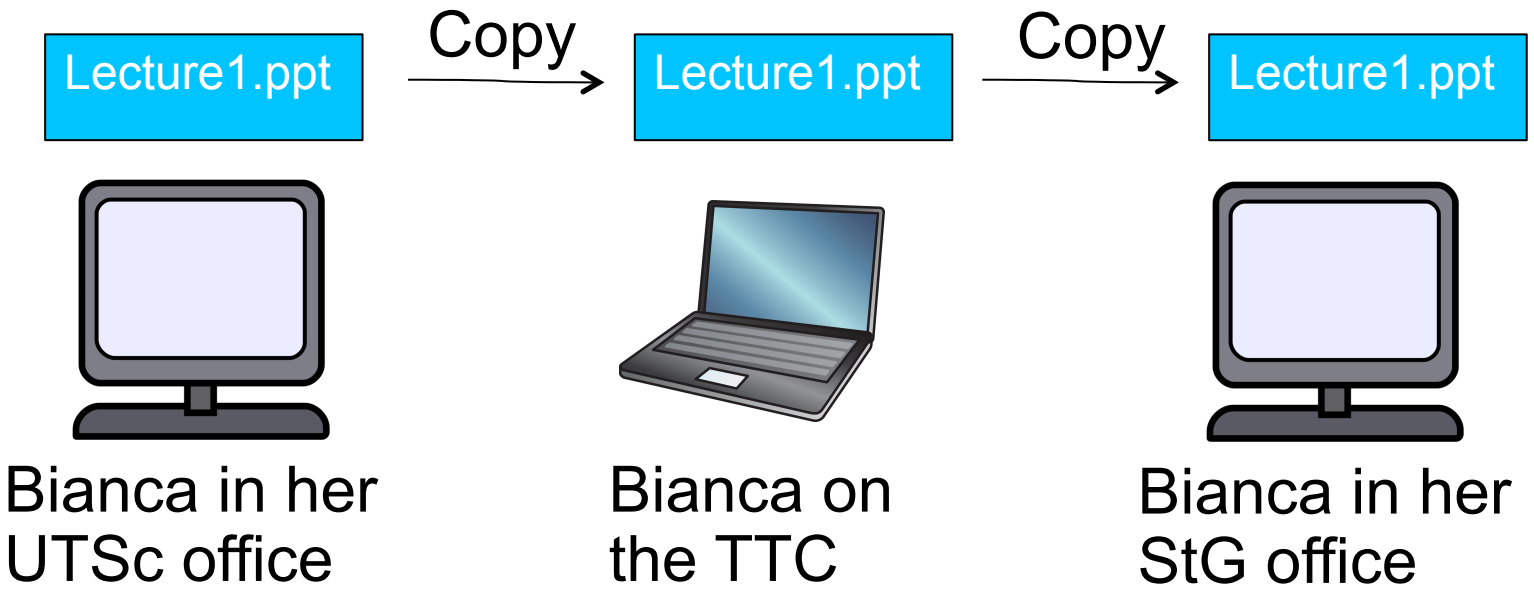
http://www.cs.toronto.edu/~bianca/cscb09s17/feedback.shtml

Assignments

- You will use a version control system (SVN) to submit your assignments.
- What is a version control system and what is it good for?

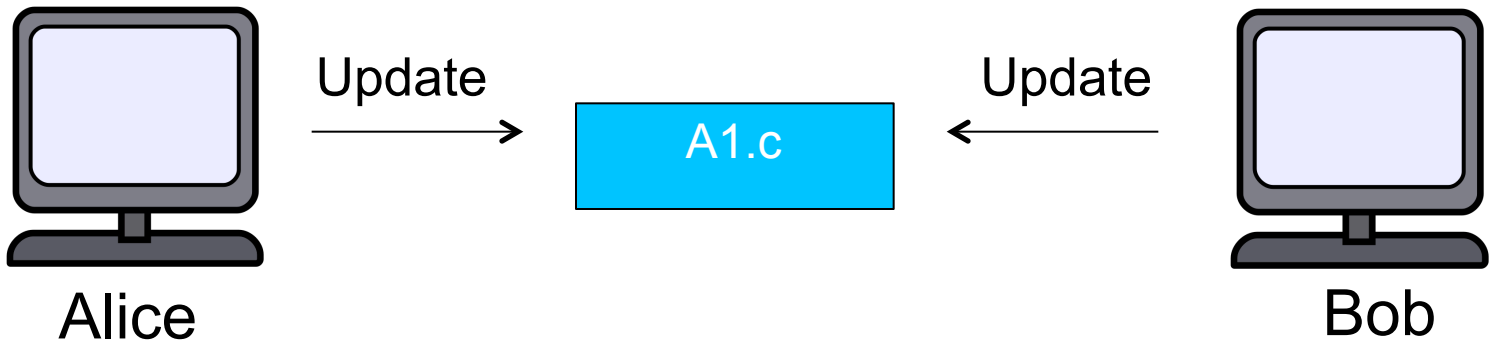
Why version control (I): Working from different machines

- Involves a lot of copying files around
- Pain to make sure to keep versions consistent



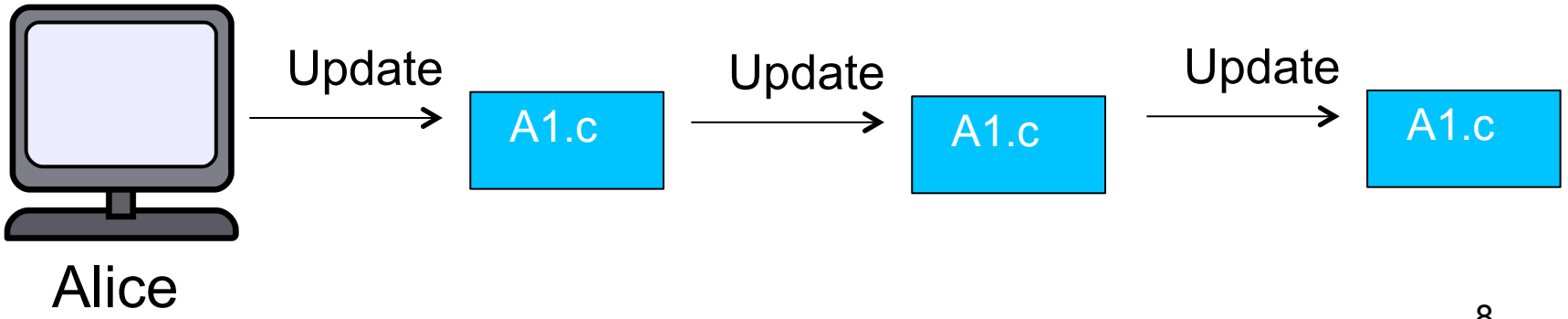
Why version control (II): Working in a team

- How do you coordinate changes to a file?
  - Exchange emails, phone calls…
  - "I'm going to work on A1.c, so don't touch it."
  - Painful!



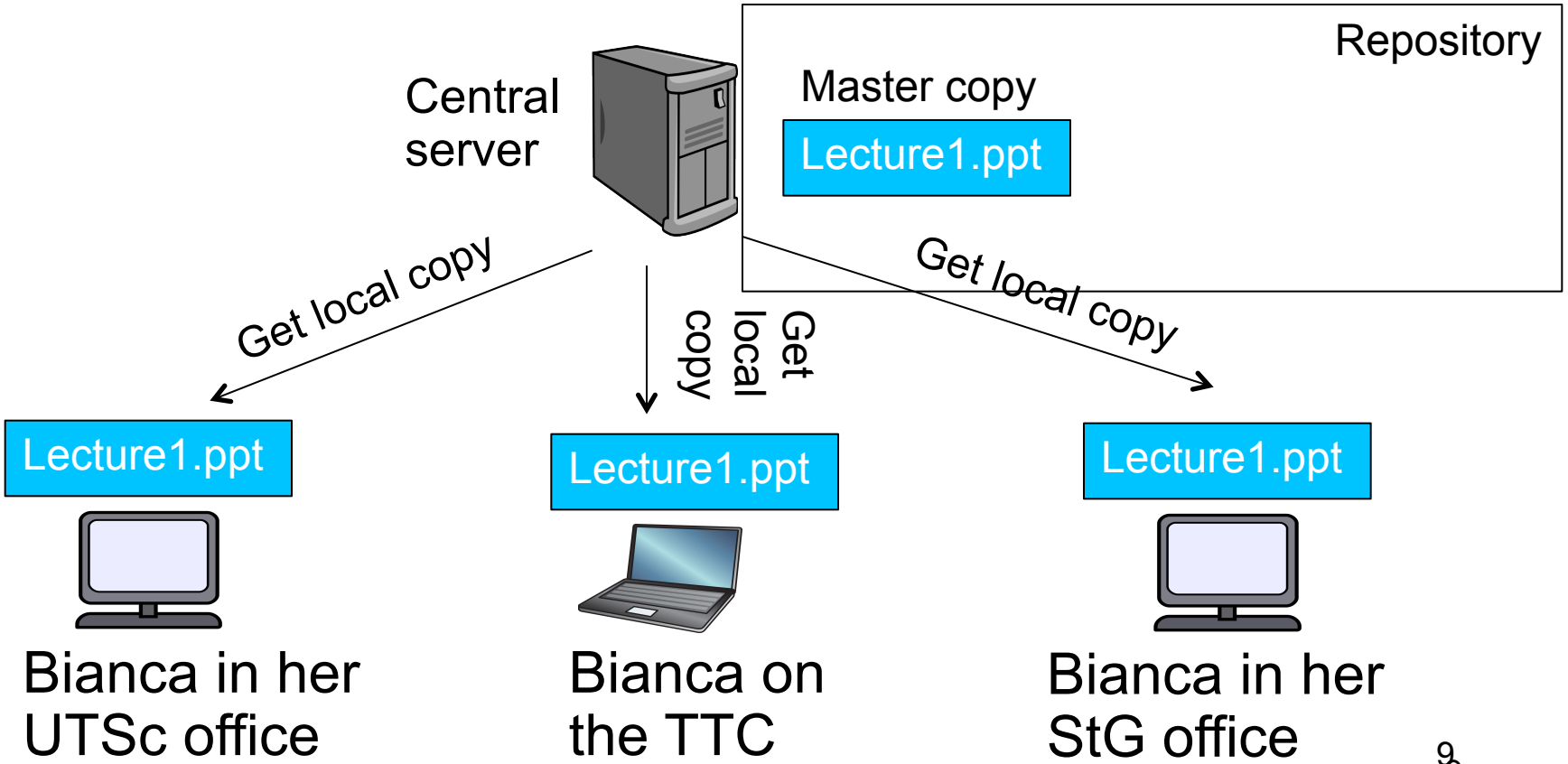
Why version control (III): Keeping track of program changes

- Could periodically save backups
  - Ad-hoc
  - Only programmers knows versions
  - Hard to pick which version to go back to
  - No tools to help you



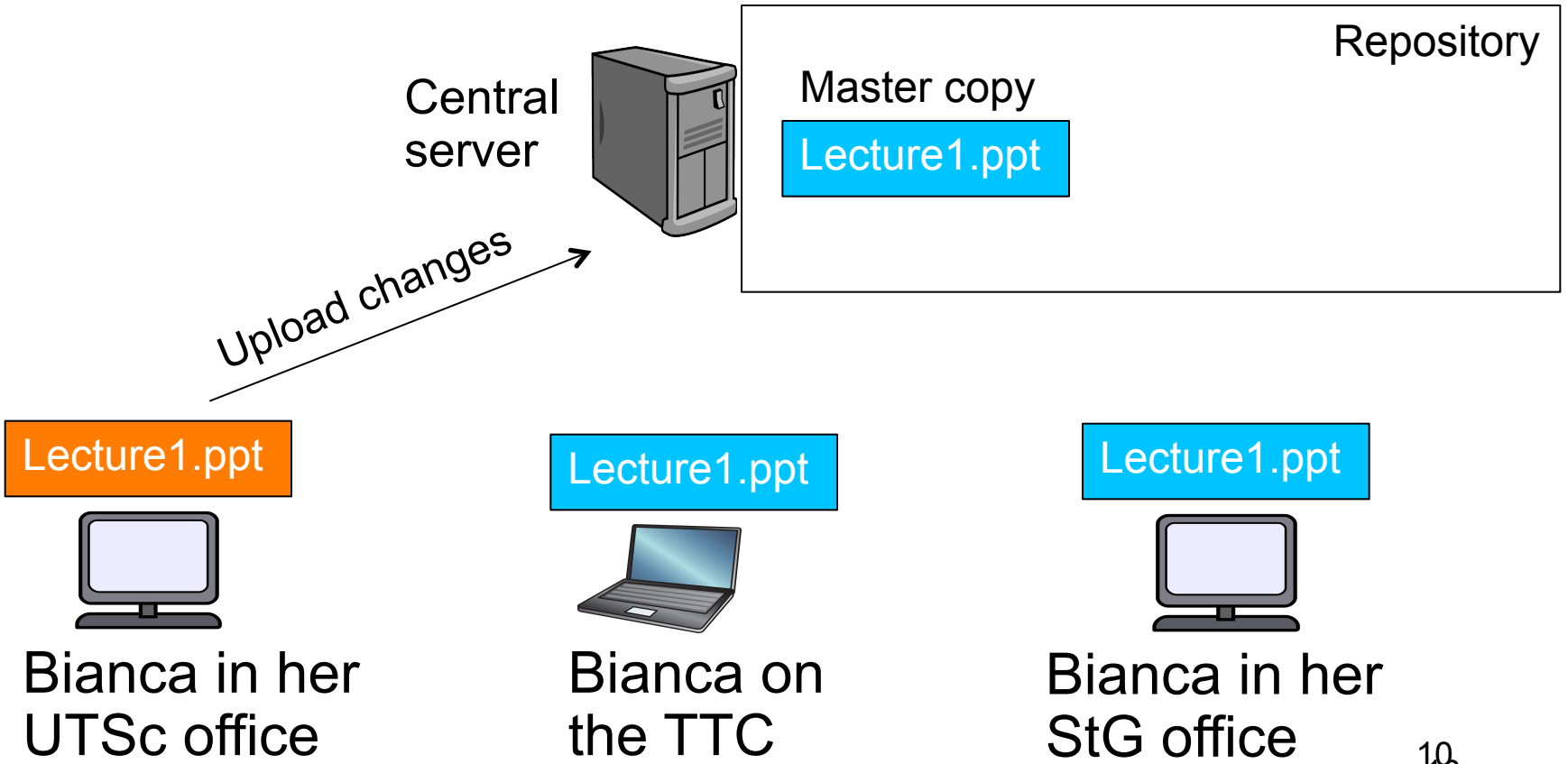
(Centralized) Version Control

- A master copy of all files (and their previous versions) lives at a central server
- For each machine / team member: before first use of repository, check out local copy
  - Under svn: `svn checkout <url of repository>`



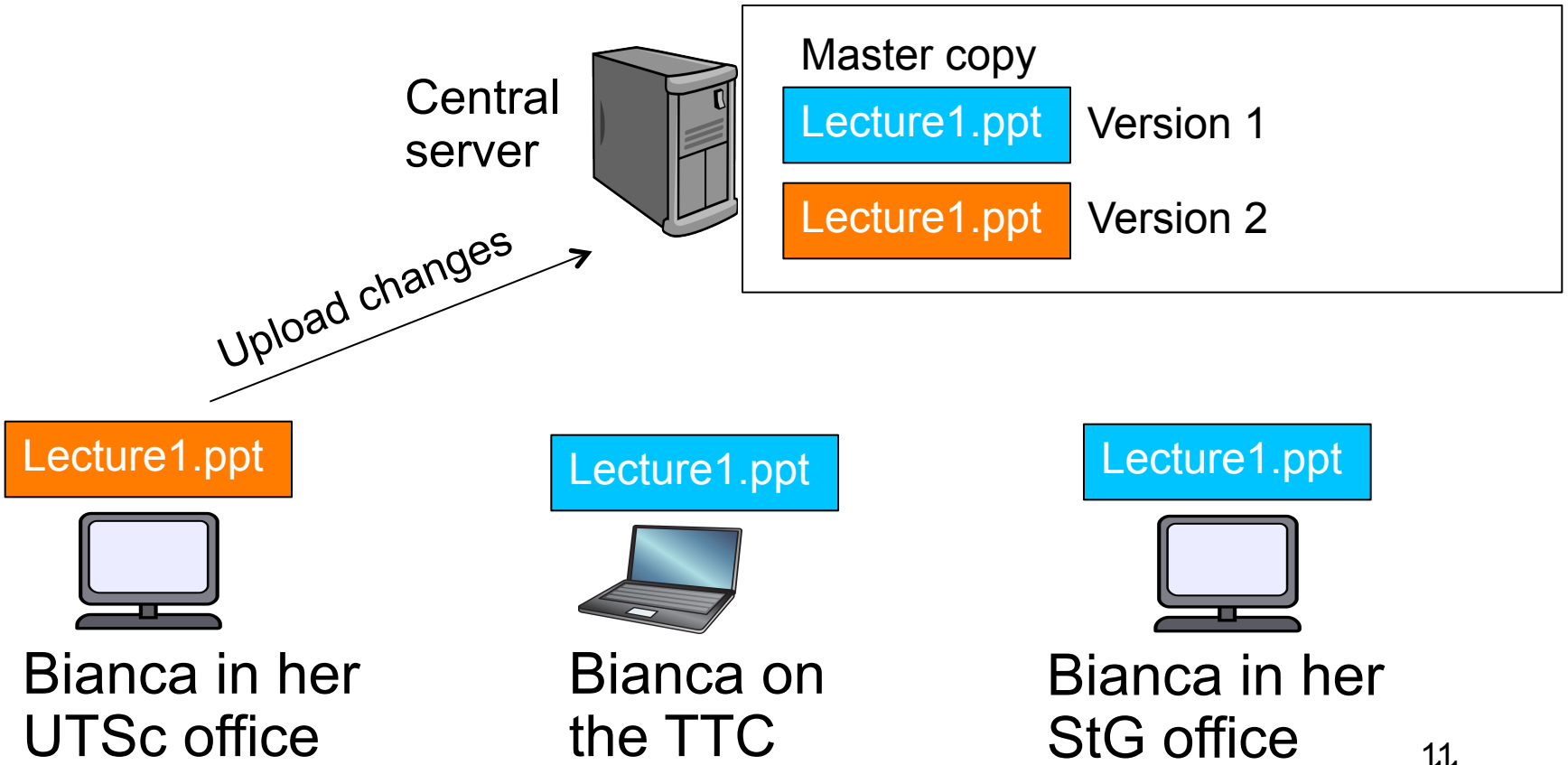
Version control

- After making modifications to local copy of file, upload changes to the server so others can see them
  - Under svn: `svn commit –m "message describing mods"`



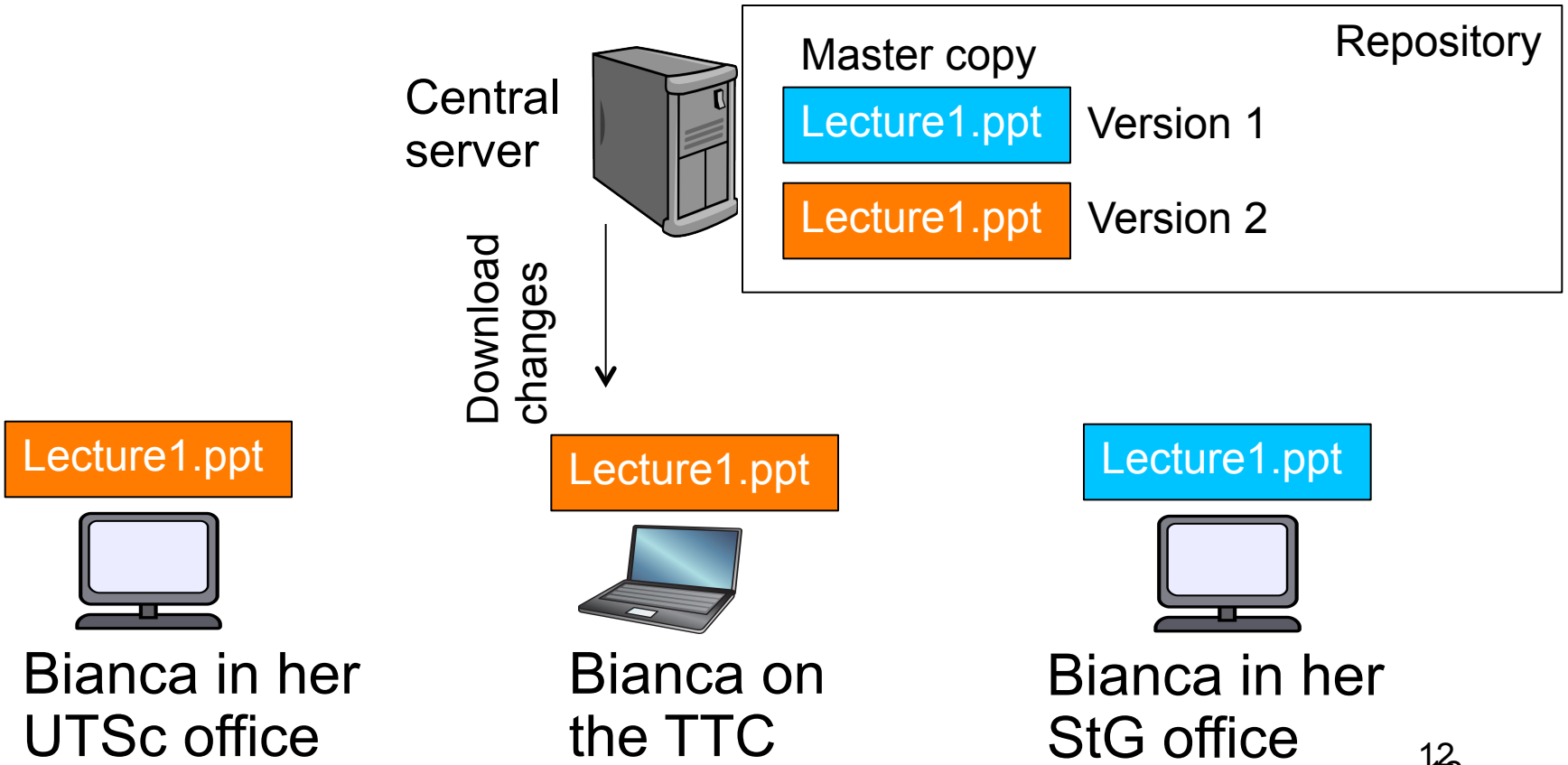
Version control

- After making modifications to local copy of file, upload changes to the server so others can see them
  - Under svn: `svn commit –m "message describing mods"`
- Note that svn stores both old and new version of file



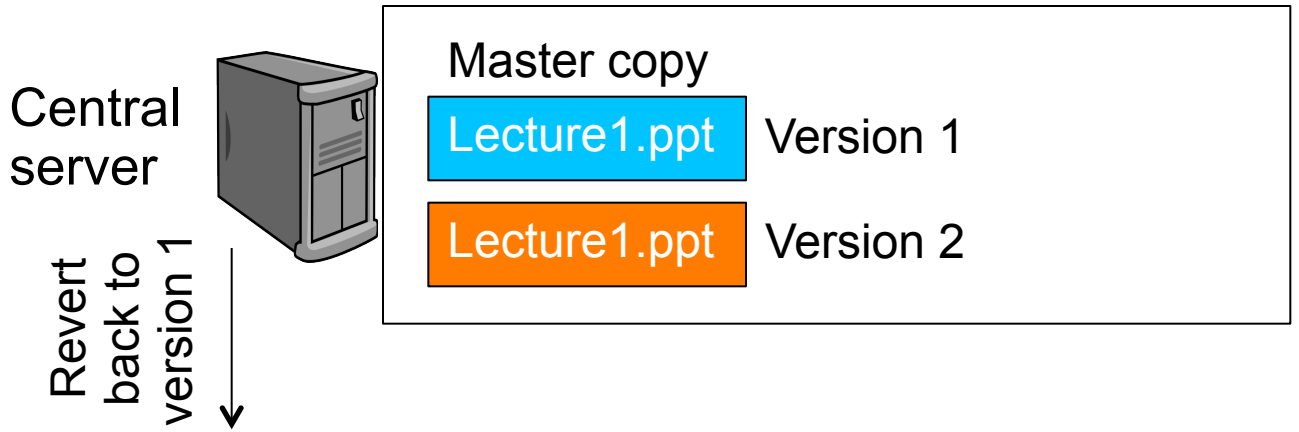
Version control

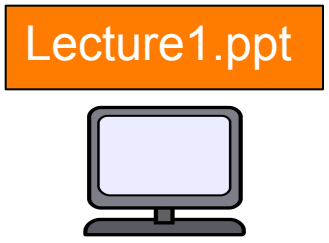
- Bianca now wants to continue working from her laptop:
  - Need to update local copy to see changes that have been "committed" to the server
  - Under svn: `svn update`



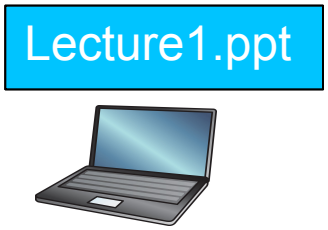
Version control

- Bianca now realizes that edits she made in the office were garbgage and wants back the original (blue) version
  - Under svn: `svn update –r 1` `(parameter to –r can be any version #)`

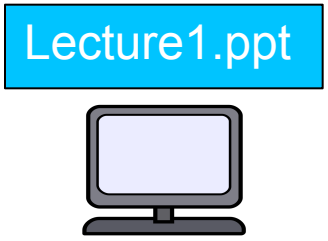




Bianca in her UTSc office



Bianca on the TTC

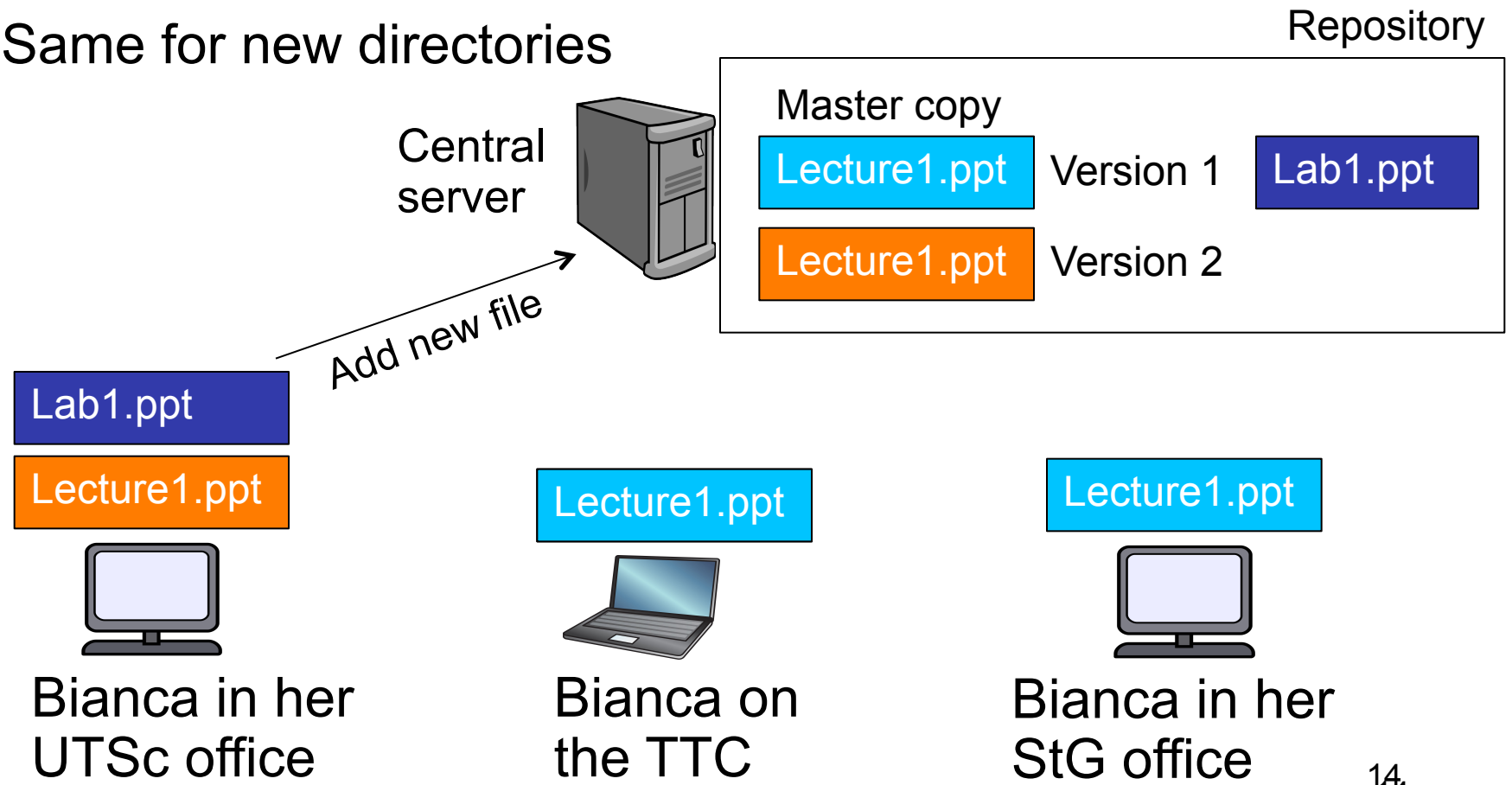


Bianca in her StG office

Version control

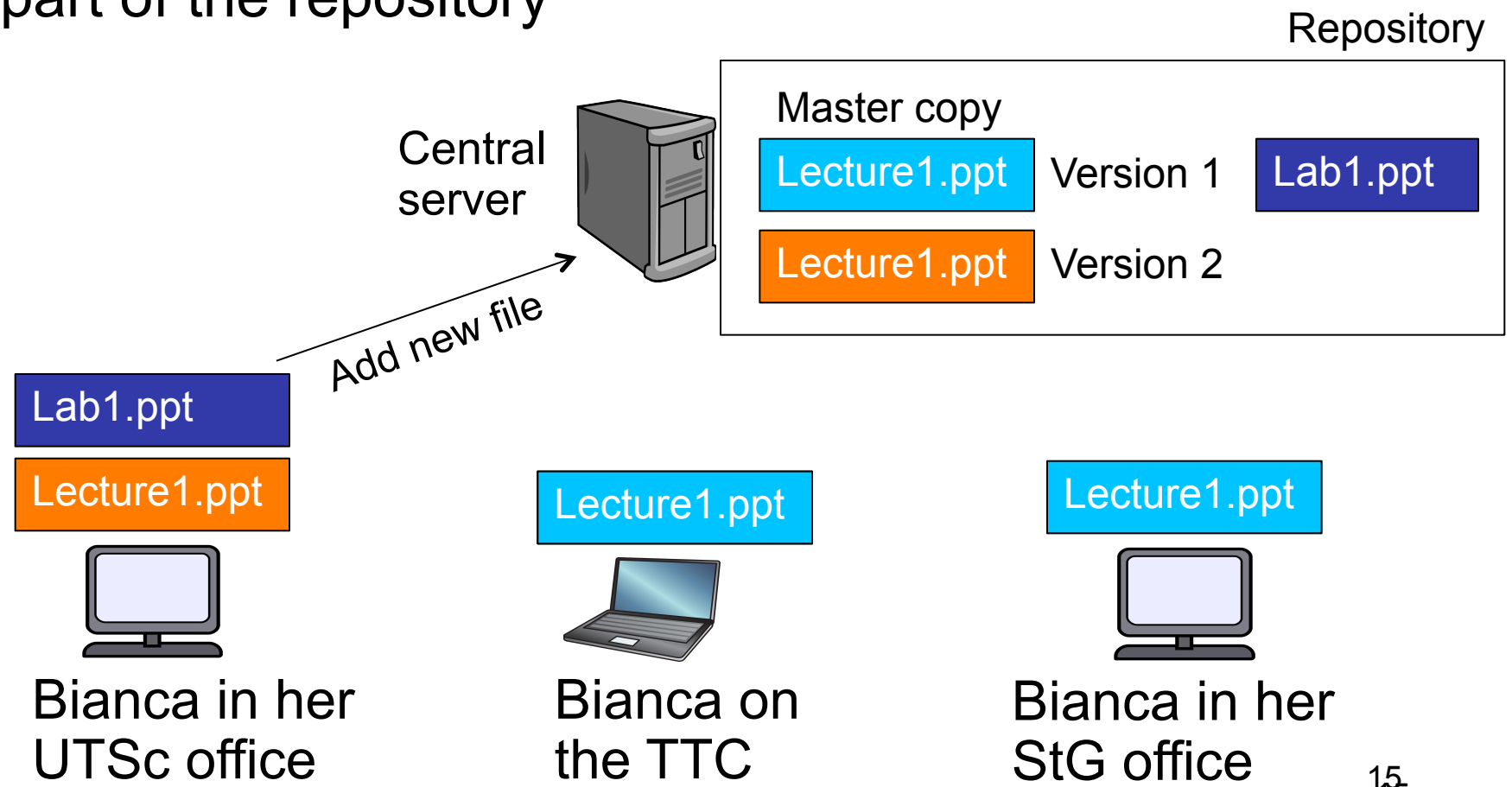
- Bianca starts working in her office on a new document
- Need to tell the server to add this file
  - Under svn: `svn add <filename>`

    `svn commit –m "message here"`
- Same for new directories



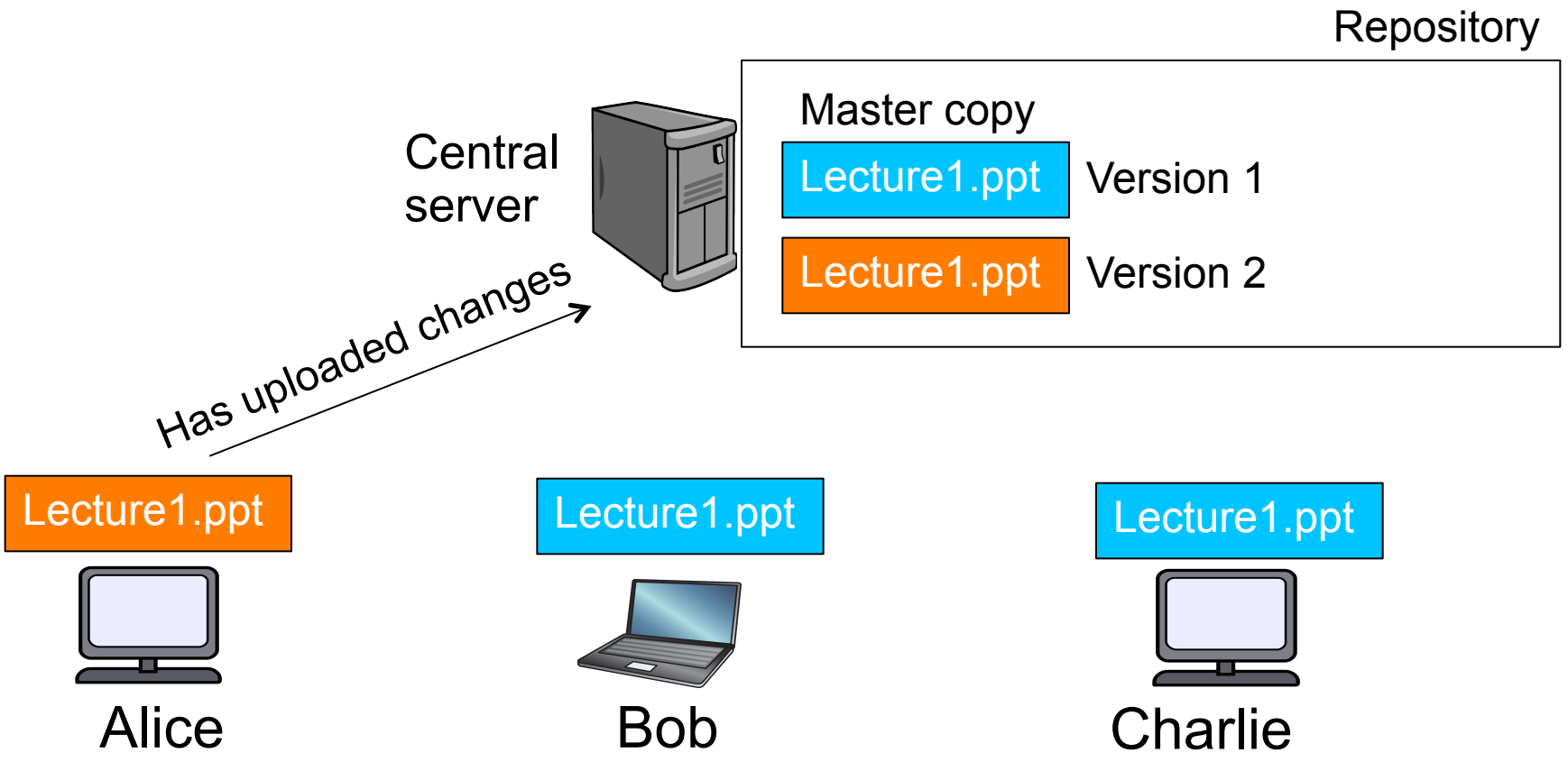
Version control

- Commit and update operations are automatically applied to all repository files in the current working directory (and recursively to sub-directories) that are part of the repository



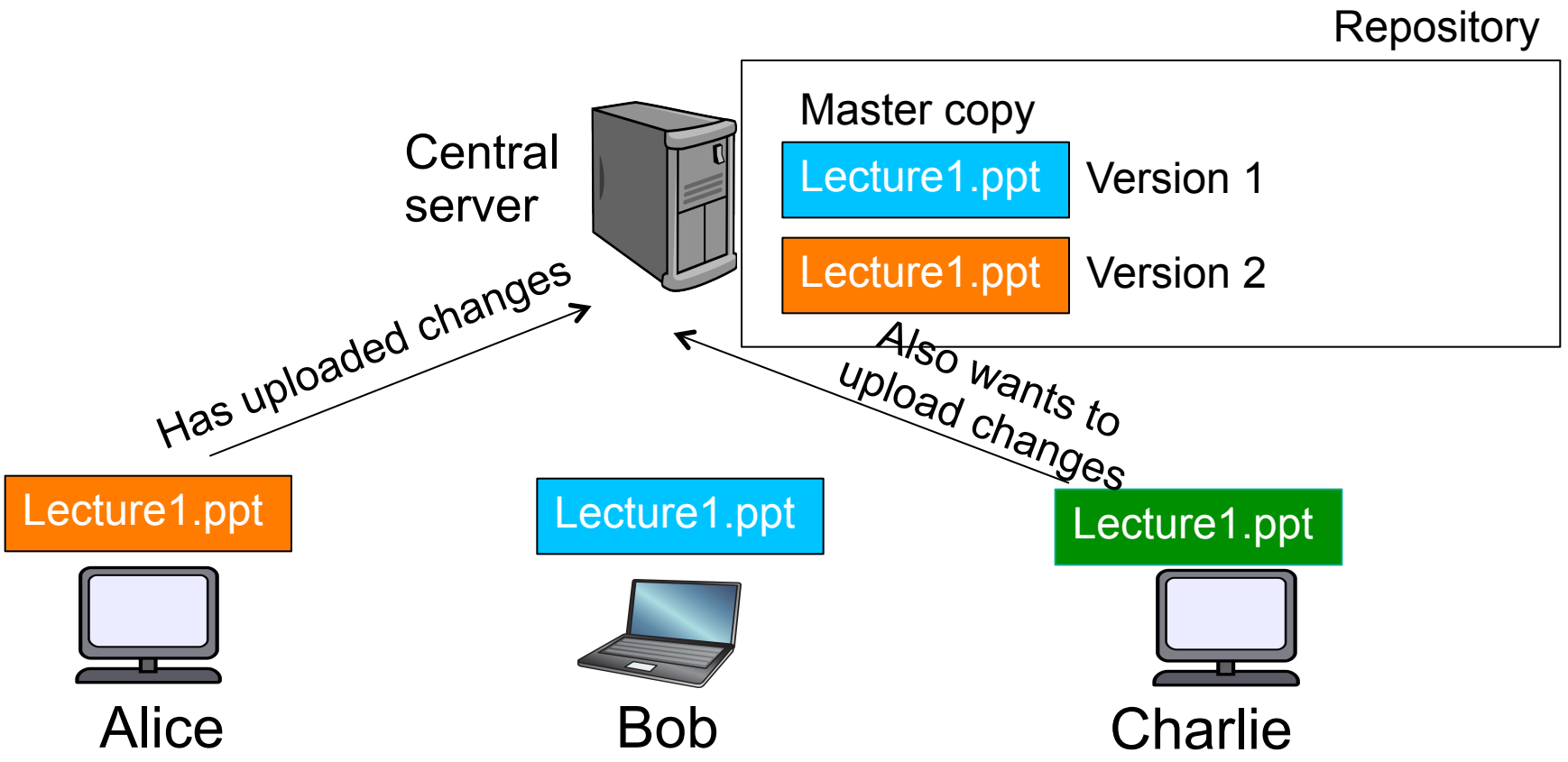
Version control

- What if two users modify their local copy, Alice uploads her changes and then Charlie tries to upload changes?
  - Will Charlie's upload overwrite Alice's changes?



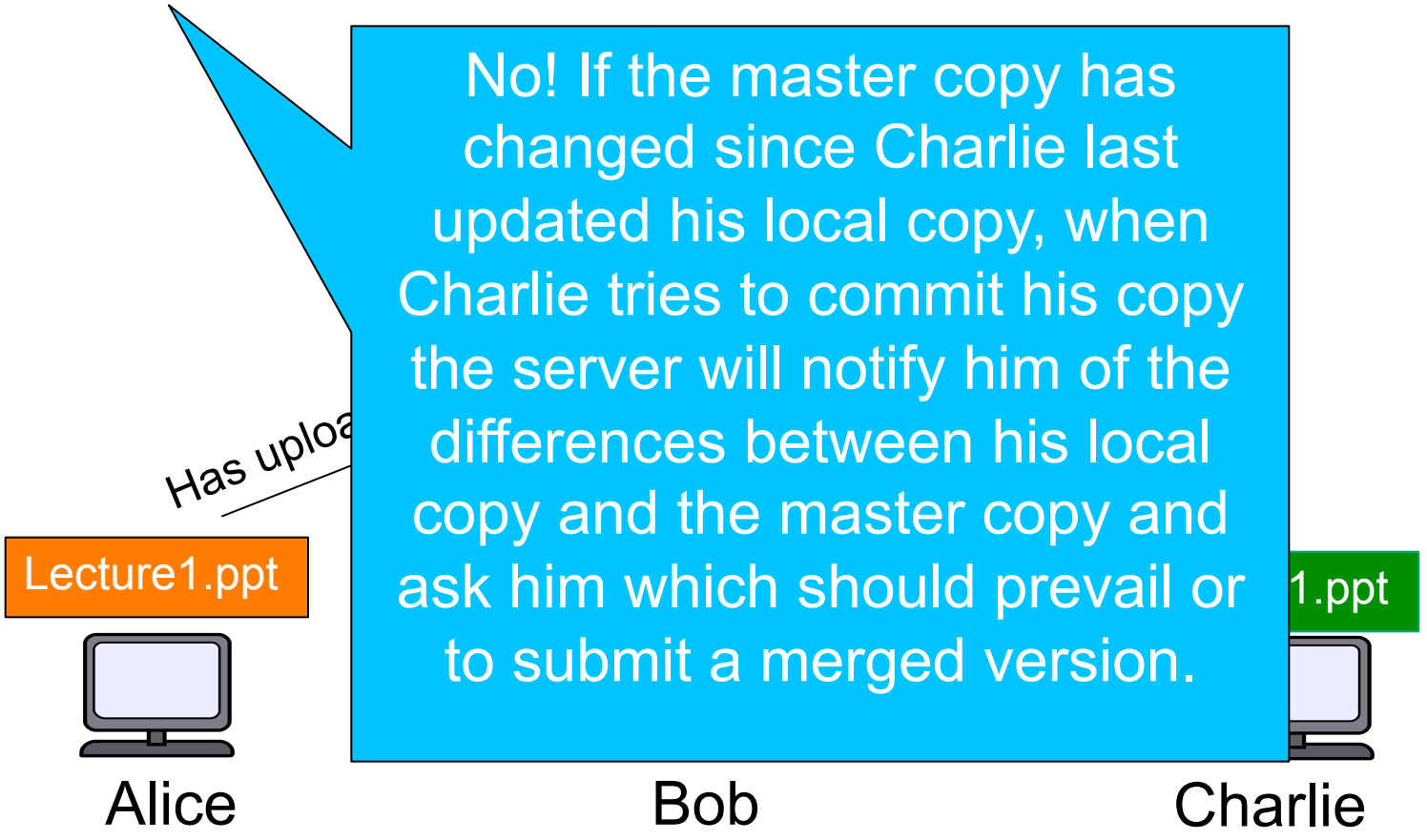
Version control

- What if two users modify their local copy, Alice uploads her changes and then Charlie tries to upload changes?
  - Will Charlie's upload overwrite Alice's changes?



Version control

- What if two users modify their local copy, Alice uploads her changes and then Charlie tries to upload changes?
  - Will Charlie's upload overwrite Alice's changes?



SVN summary

- Checking out a new SVN repository:
  - `svn checkout <url of repository>`
- Committing local changes to the repository:
  - `svn commit –m “message describing mods”`
- Downloading updates in the repository to local copy:
  - `svn update`
- Adding new files/directories to the repository:
  - `svn add <filename>`

Why should you learn C?

The Tiobe index for the popularity of programming languages:

May 2017	May 2016	Change	Programming Language	Ratings
1	1		Java	14.639%
2	2		C	7.002%
3	3		C++	4.751%
4	5	^	Python	3.548%
5	4	v	C#	3.457%
6	10	^^	Visual Basic .NET	3.391%
7	7		JavaScript	3.071%

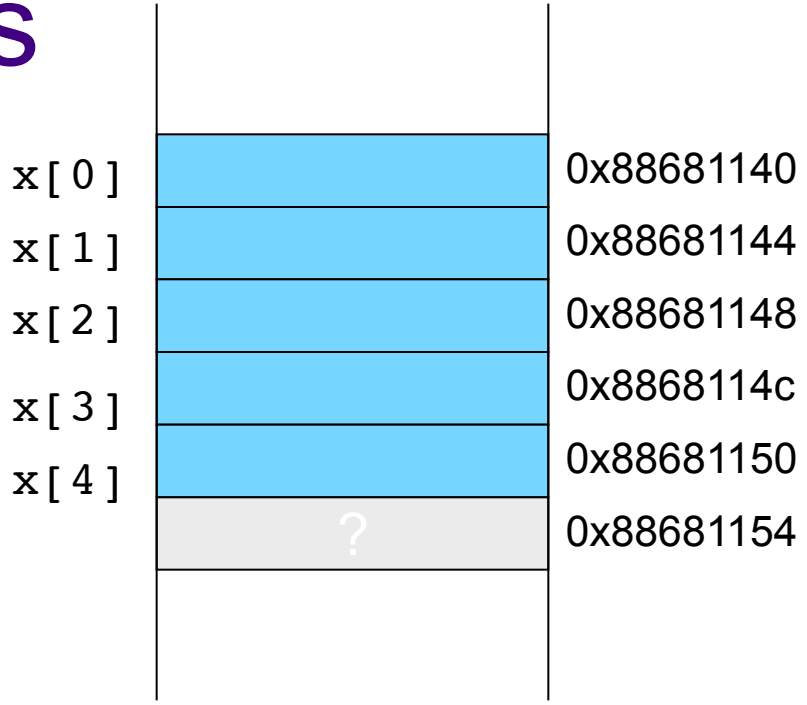
- Also required in many of your future courses (operating systems, advanced databases, compilers, graphics..)

A sample program

```
#include <stdio.h>
#define DAYS 4

int main() {

    float daytime_high[DAYS] = {16.0, 12.8, 14.6, 19.1};

    float average_temp = 0;

    int i;
    for (i = 0; i < DAYS; i++) {
         average_temp += daytime_high[i];
    }

    average_temp = average_temp / DAYS;
    printf("average %f\n", average_temp);

    return 0;
}
```

Arrays

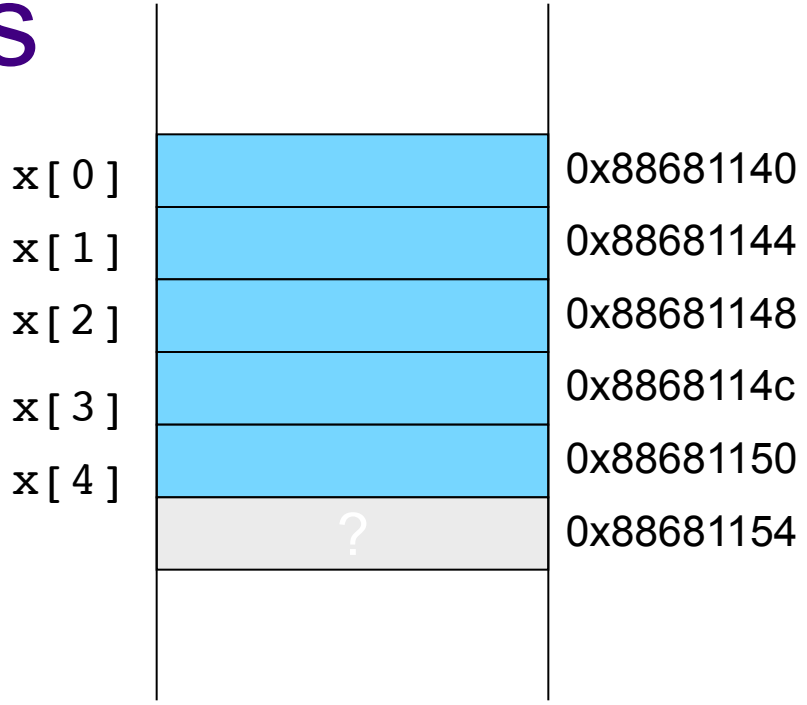
```
int x[5];
for (i = 0; i < 5; i++) {
    x[i] = i*i;
}
```



- Arrays in C are a contiguous chunk of memory that contain a list of items of the same type.
- If an array of ints contains 10 ints, then the array is 40 bytes (assuming 4 byte integers). There is nothing extra.
- In particular, the size of the array is not stored with the array. There is *no* runtime checking.

Arrays

```
int x[5];
for (i = 0; i <= 5; i++) {
    x[i] = i*i;
}
```



- No runtime checking of array bounds
- Behaviour of exceeding array bounds is “undefined”
  - → program might appear to work
  - → program might crash
  - → program might do something apparently random

Number representations

- Counting in the decimal system:

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 …

Represents numbers as powers of 10, e.g. the number 1315 equals:

1 x 1000 + 3 x 100 + 1 x 10 + 5 x 1

- Counting in the hexadecimal system:

0 1 2 3 4 5 6 7 8 9 A B C D E F 10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F 20

Represents numbers as powers of 16, e.g. the number 1315 equals:

1 x 16^3 + 3 x 16^2 + 1 x 16^1 + 5 x 16^0