Version Control

Naisan Benatar

On today's menu…

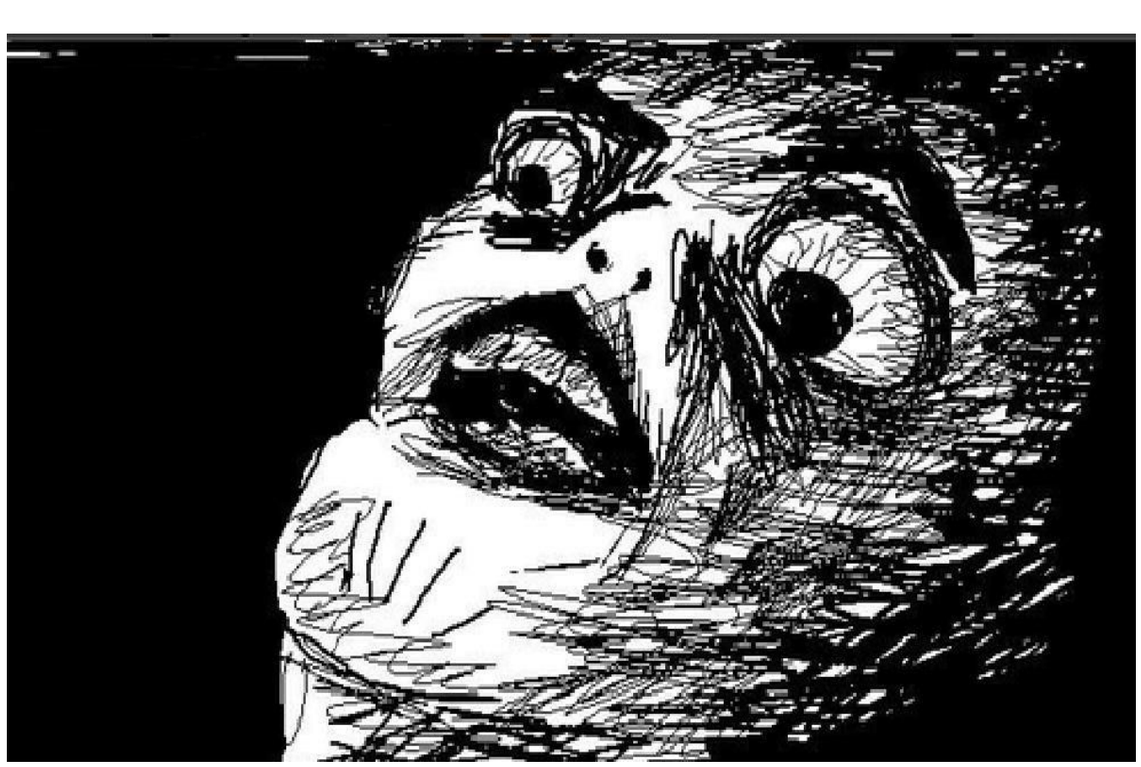
- The problems with lots of code and lots of people
- Version control systems
  - what are they?
  - how are they used?
  - centralised versus distributed version control
  - Features of version control including branching

Dealing with Change

- How do you manage your coursework?
  - Modifying existing code (using Q1 for a basis for Q2)
  - Backing up working code
  - Checking if an idea works (Do I use a Hashtable or a HashMap?)
  - Sharing code in group projects

(Bad) Solutions

- Copying (Coursework_working.java, Coursework_tmp.java)
- Copy & Paste code snippets
- Copy entire directories
- Emailing code to people

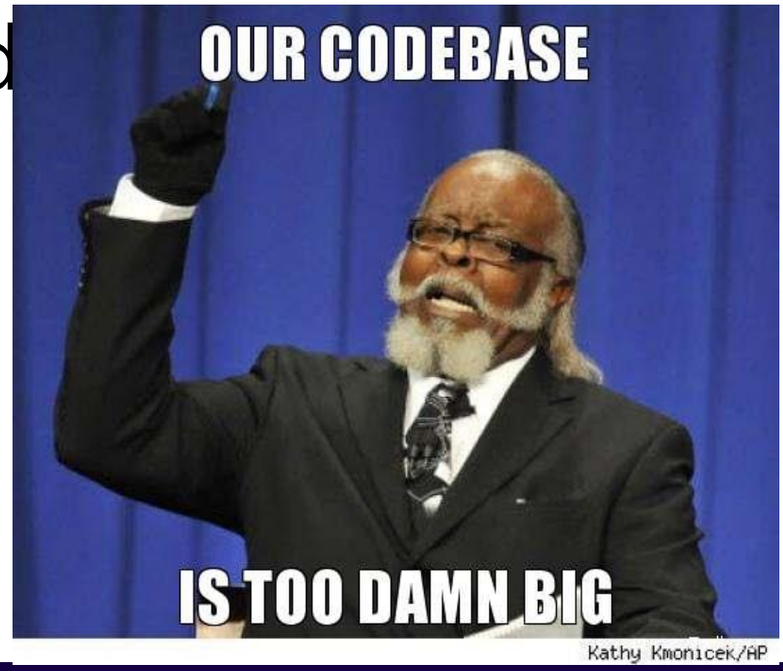


Open Source

- You thought coursework was bad?
- Linux kernel has thousands of regular developers, millions of files.
- Developers spread over the globe across multiple time zones

Big code bases

- Operating systems code
  - Win 95 approx 5 million lines of code (1995)
  - Linux kernel 2.6.37 14 million lines of code (2011)
- Modern PC game
  - Unreal 3 approx 500,000 lines of cod

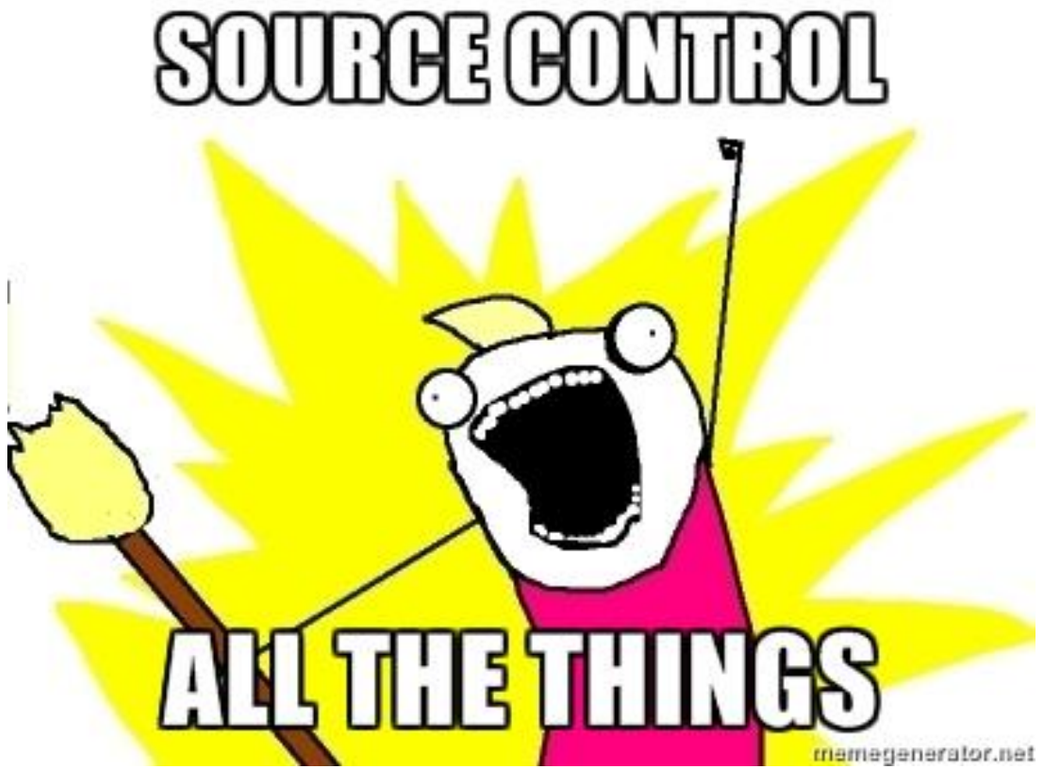


Making a mess

- The Linux kernel runs on different processors (ARM, x86, MIPS). These can require significant differences in low level parts of the code base
- Many different modules
- Old versions are required for legacy systems
- Because it is open source, any one can download and suggest changes.
- How can we create a single kernel from all of this?

Not just code!

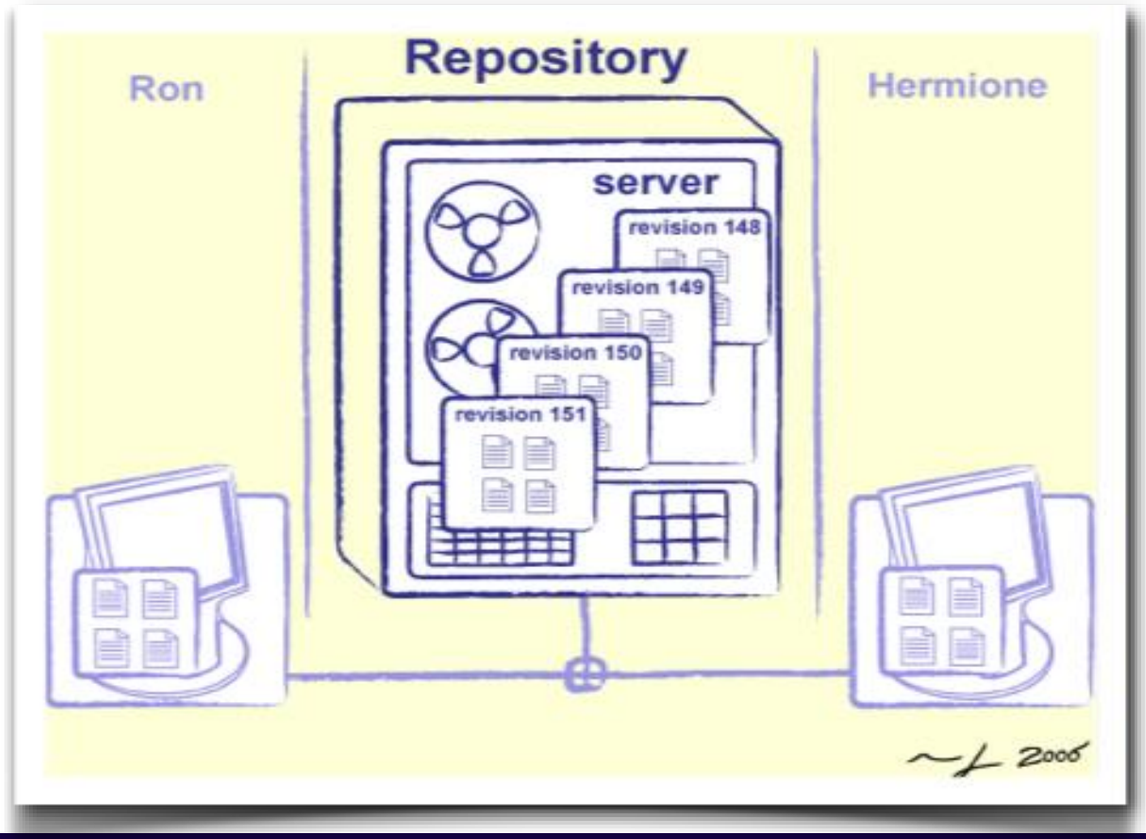
- A *Code Base* does not just mean code!
- Also includes:
  - Documentation
  - Build Tools (Makefiles etc)
  - Configuration files
- But NOT a certain type of file




DEVELOPERS
Y U NO USE SOURCE CONTROL
memegenerator.net

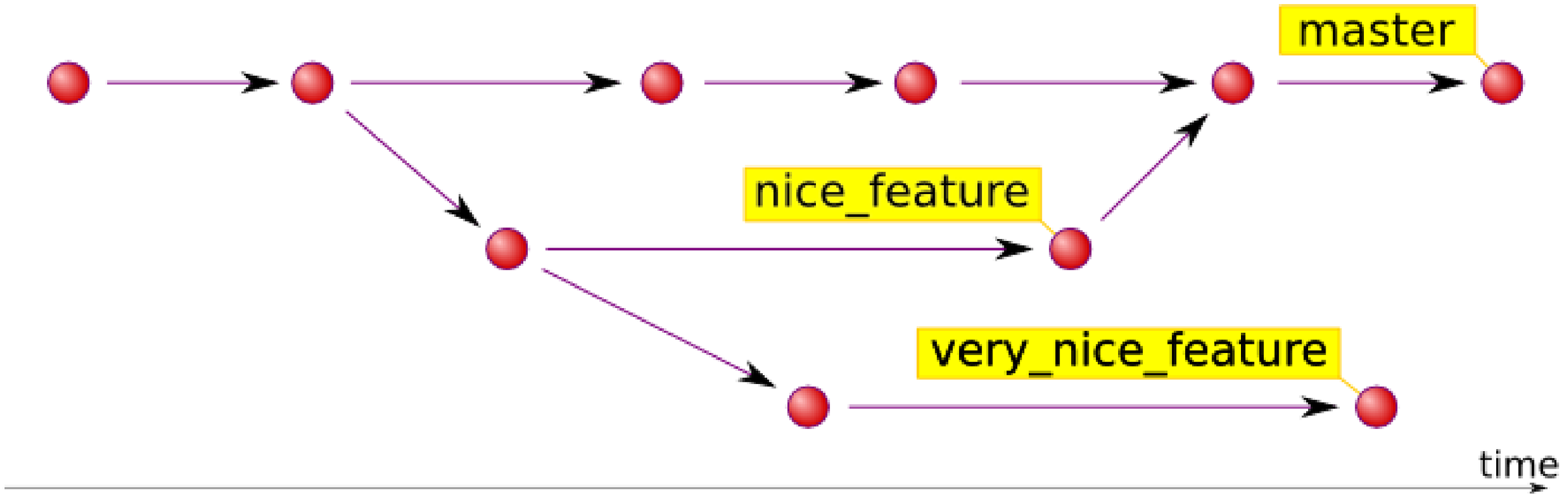
Control the process automatically

- Manage these things using a version control system (VCS)
- A version control system is a system which allows for the management of a code base.



Details of the process

- Files are kept in a *repository*
- Repositories can be local or remote to the user
- The user edits a copy called the *working copy*
- Changes are *committed* to the repository when the user is finished making changes
- Other people can then access the repository to get the new code
- Can also be used to manage files when working across multiple computers

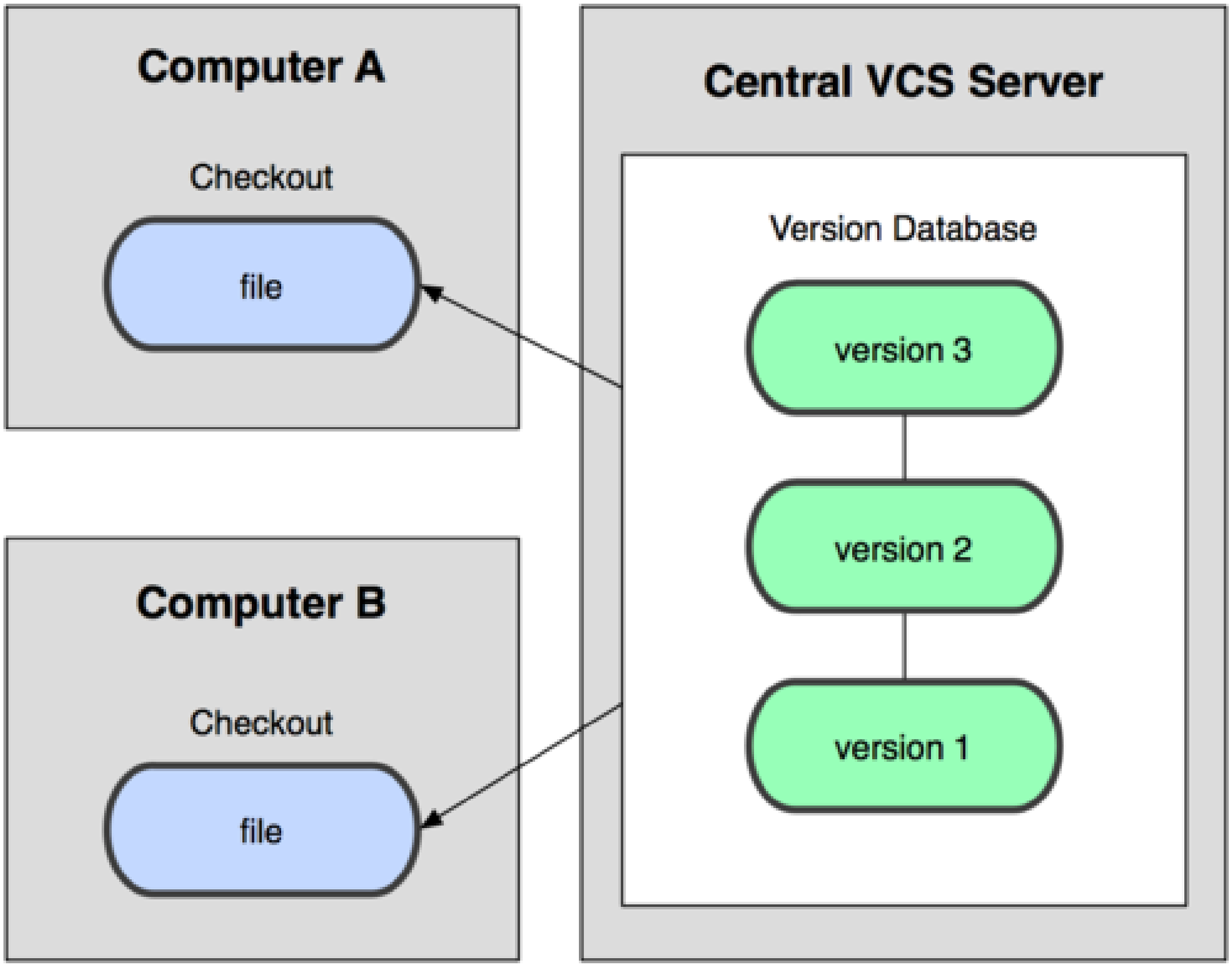

master
nice_feature
very_nice_feature
time

Centralised Version Control

- A single server holds the code base
- Clients access the server by means of check-in/check-outs
- Examples include CVS, Subversion, Visual Source Safe.

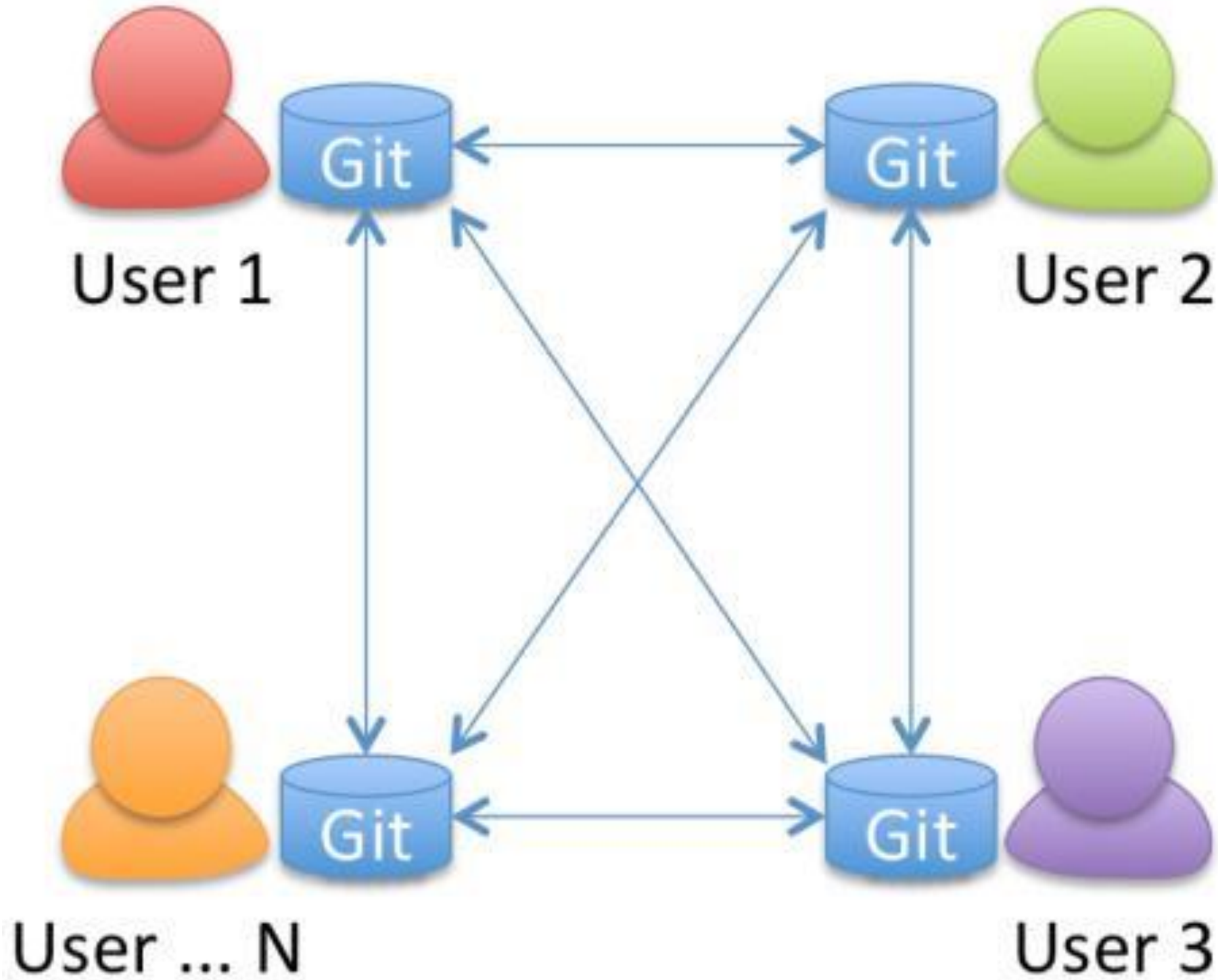
Advantages: Easier to maintain a single server.

Disadvantages: Single point of failure.


Computer A
Checkout
file
Computer B
Checkout
file
Central VCS Server
Version Database
version 3
version 2
version 1

Distributed Version Control

- Each client (essentially) holds a complete copy of the code base.
- Code is shared between clients by push/pulls
  - Advantages: Many operations cheaper. No single point of failure
  - Disadvantages: A bit more complicated!


Git
Git
User 1
User 2
Git
Git
User ... N
User 3

More Uses of Version Control

- Version control is not just useful for collaborative working, essential for quality source code development
- Often want to undo changes to a file
  - start work, realize it's the wrong approach, want to get back to starting point
  - like "undo" in an editor…
  - keep the whole history of every file and a *changelog*
- Also want to be able to see who changed what, when
  - The best way to find out how something works is often to ask the person who wrote it

Branching

- Branches allows multiple copies of the code base within a single repository.
  - Different customers have different requirements
    - Customer A wants features A,B, C
    - Customer B wants features A & C but not B because his computer is old and it slows down too much.
    - Customer C wants only feature A due to costs
  - Each customer has their own branch.
- Different versions can easily be maintained

Selecting a VCS

- When choosing a VCS consider:
  - How many files and developers are likely to be involved in the project?
  - Speed for common operations (check-in, check-out)
  - Is there a server? Does it need to be powerful?

Essential features

- Check-in and check-out of items to repository
- Creation of baselines (labels/tags)
  - Version 1.0 released!
- Control and manipulation of branching
  - management of multiple versions
- Overview of version history

Additional Features (1)

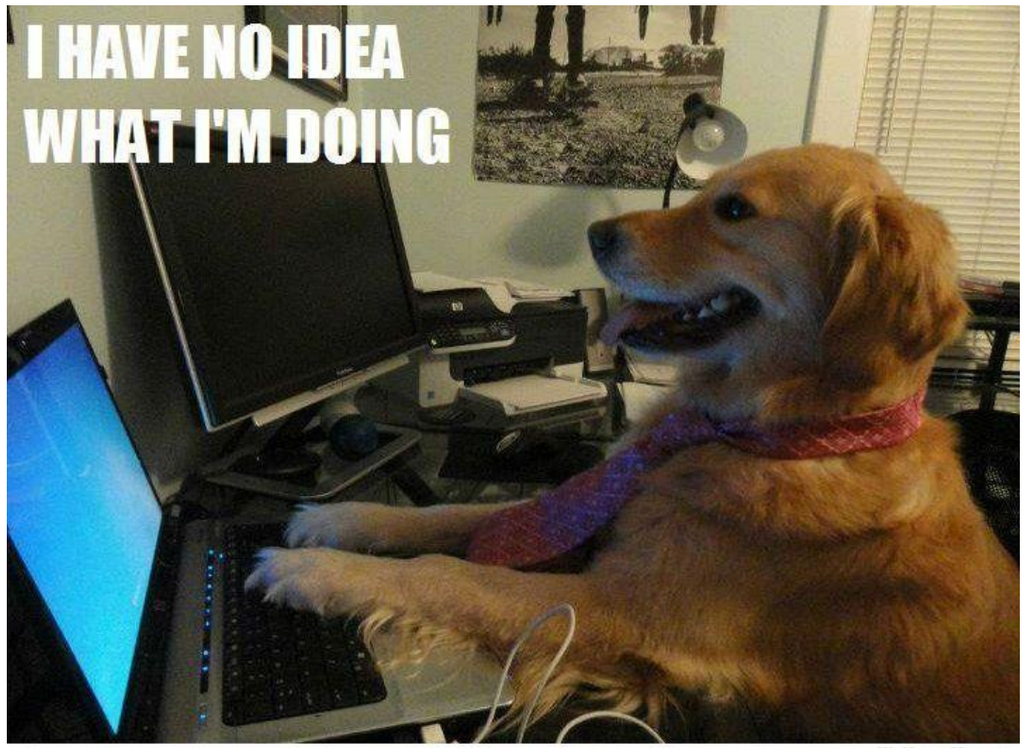
- *Change Management:*
  - Professional software will have bugs. Customers will find them. How do we know if a bug has been fixed?
  - Check-outs of code usually controlled.
    - A bug report will identify where the bug is in the code.
    - The fixed code (patch) is checked in and linked to bug report
    - Hence we can see exactly what changes were made in response to a specific bug. Good for accountability

Additional Features (2)

- Code responsibility & Code audits.
  - You stole my code!
  - Who is responsible for this module?
  - Legal stuff
- Forking – Common with Open source software
  - A subset of developers *fork* off a parent project to produce a second copy of the project. Reasons vary but often done to make a more specific version.
- Metrics (Managers only!)

Check Outs

- If you want to make a change the file needs to be *checked out* from the repository
- Usually done a file at a time.
- Some VCSs will lock checked out files so only one person may edit at a time.

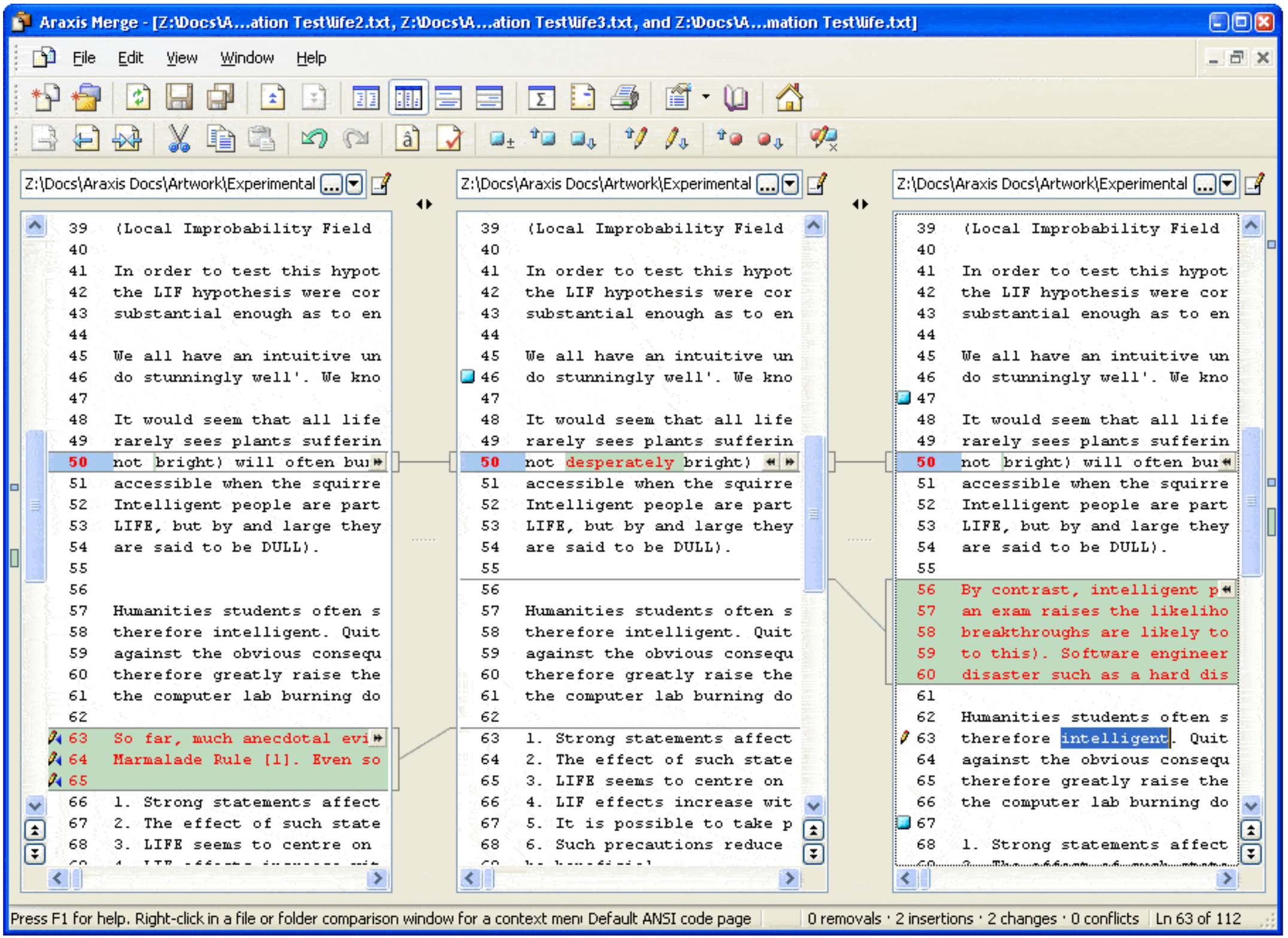


Check-In

- When changes are completed the new code is *checked-in*.
- A *commit* consists of a set of checked in files and the diff between the new and parent versions of each file.
- Each check-in is accompanied by a user name and other meta data.
- Check-ins can be exported from the Version Control system the form of a *patch*.

Merging

- There are occasions when multiple versions of a file need to be collapsed into a single version.
  - E.g. A feature from one branch is required in another
- This process is known as a merge.
- Difficult and dangerous to do in CVS
- Easy and cheap to do it git


Araxis Merge - [Z:\Docs\A...ation TestWife2.txt, Z:\Docs\A...ation TestWife3.txt, and Z:\Docs\A...mation TestWife.txt]
File Edit View Window Help
Z:\Docs\Araxis Docs\Artwork\Experimental
Z:\Docs\Araxis Docs\Artwork\Experimental
Z:\Docs\Araxis Docs\Artwork\Experimental
39 (Local Improbability Field
40
41 In order to test this hypot
42 the LIF hypothesis were cor
43 substantial enough as to en
44
45 We all have an intuitive un
46 do stunningly well'. We kno
47
48 It would seem that all life
49 rarely sees plants sufferin
50 not bright) will often bu
51 accessible when the squirre
52 Intelligent people are part
53 LIFE, but by and large they
54 are said to be DULL).
55
56
57 Humanities students often s
58 therefore intelligent. Quit
59 against the obvious consequ
60 therefore greatly raise the
61 the computer lab burning do
62
63 So far, much anecdotal evi
64 Marmalade Rule [1]. Even so
65
66 1. Strong statements affect
67 2. The effect of such state
68 3. LIFE seems to centre on
39 (Local Improbability Field
40
41 In order to test this hypot
42 the LIF hypothesis were cor
43 substantial enough as to en
44
45 We all have an intuitive un
46 do stunningly well'. We kno
47
48 It would seem that all life
49 rarely sees plants sufferin
50 not desperately bright)
51 accessible when the squirre
52 Intelligent people are part
53 LIFE, but by and large they
54 are said to be DULL).
55
56
57 Humanities students often s
58 therefore intelligent. Quit
59 against the obvious consequ
60 therefore greatly raise the
61 the computer lab burning do
62
63 1. Strong statements affect
64 2. The effect of such state
65 3. LIFE seems to centre on
66 4. LIF effects increase wit
67 5. It is possible to take p
68 6. Such precautions reduce
39 (Local Improbability Field
40
41 In order to test this hypot
42 the LIF hypothesis were cor
43 substantial enough as to en
44
45 We all have an intuitive un
46 do stunningly well'. We kno
47
48 It would seem that all life
49 rarely sees plants sufferin
50 not bright) will often bu
51 accessible when the squirre
52 Intelligent people are part
53 LIFE, but by and large they
54 are said to be DULL).
55
56 By contrast, intelligent p
57 an exam raises the likeliho
58 breakthroughs are likely to
59 to this). Software engineer
60 disaster such as a hard dis
61
62 Humanities students often s
63 therefore intelligent. Quit
64 against the obvious consequ
65 therefore greatly raise the
66 the computer lab burning do
67
68 1. Strong statements affect
Press F1 for help. Right-click in a file or folder comparison window for a context menu Default ANSI code page
0 removals · 2 insertions · 2 changes · 0 conflicts
Ln 63 of 112

Version Control in action

- I use git for my day to day work. One developer, lots of code written over 3 years in multiple languages (C, Python, Java, shell, awk)
- I need a regular back-up system but work on at least 3 PCs (Home, work and work linux).

Using a git repository

- `$ git init`
- `$ git add <filename>`
- `$ git commit -a`
- `$ git branch`
- `$ git checkout`
- `$ git push`
- `$ git pull`