Git Branching & Conflicts

Welcome to the New Collaboration!

Git What?

- If you have not done so already, review the descriptions, materials, and video detailed in the Introduction to Git portion of section 1.8 Git and GitHub on Blackboard.
- You will gain an understanding of Git, version control, and why it’s important.

More Background

- Review Chapter 1 Getting Started
- Review Section 1.1 About Version Control
- Review Section 1.2 A Short History of Git
- Review Section 1.3 Git Basics
- You can navigate these sections using the “Prev” and “Next” links at the bottom of the page.

Recapping 1a

- You read background about Git.
- You installed Git on your local machine.
- You configured Git with your name, email, the Command Line editor Vim, and a conflicts tool.
- You created a new folder for your work called JointProject and launched the Git processes and code for that folder.
- You checked the Git status of your folder using “git status”.

Recapping 1a

- You added a file called README to your folder using Vim.
- You discovered that the file was untracked.
- You tracked the file using “git add”, turning it into a “staged” file.
- You created a snapshot of the project status using “git commit” and described your changes using Vim.
- You checked your progress and commit comment using “git log”.

Pro Git Book

- For this exercise, we will be working through portions of the third section of the *Pro Git Book*, written by Scott Chacon and published by Apress.
- Browse to the table of contents for *Pro Git* at http://git-scm.com/book.
- We will be working in the Branching chapter.

Branching Background

- Review Chapter 3 Git Branching
- Review Section 3.1 What a Branch Is
- You can navigate these sections using the “Prev” and “Next” links at the bottom of the page.

Git Branching

Let’s get started!

Back to Command Line

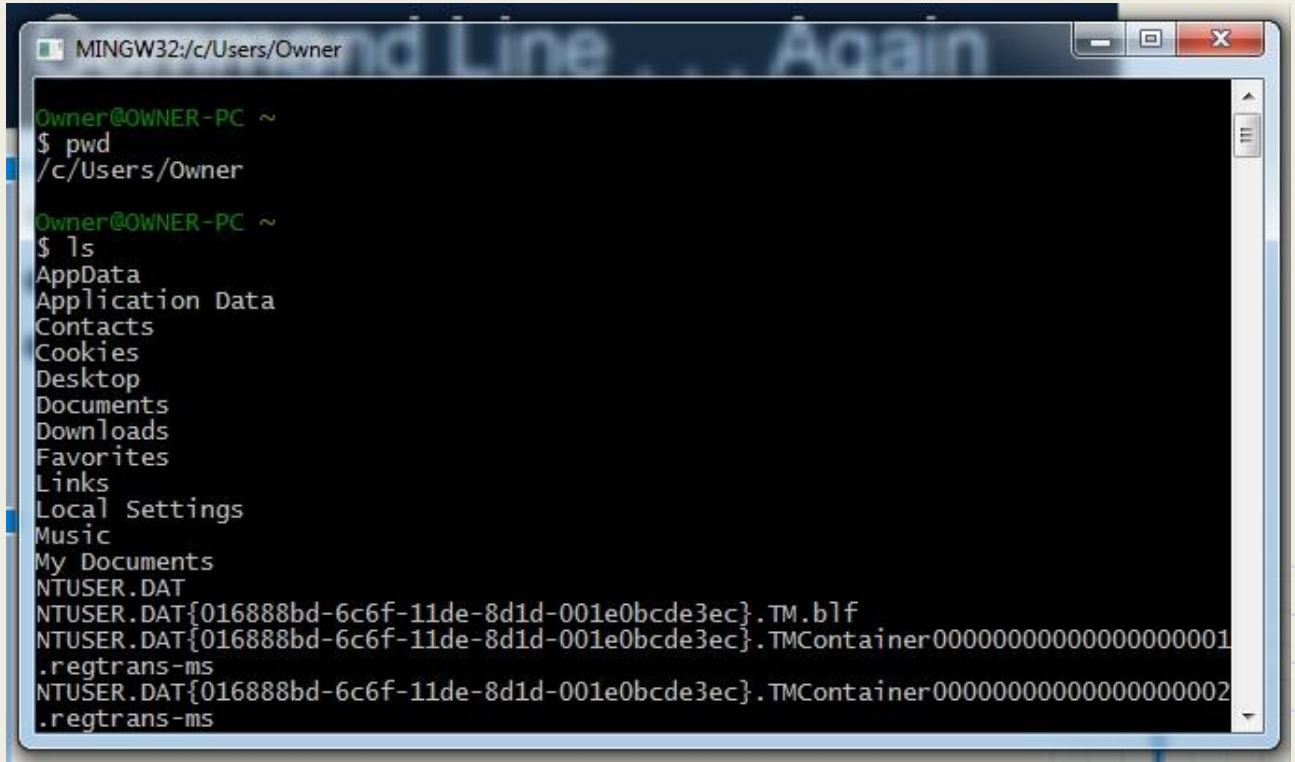
- Start up the Git agent. It’s called Git Bash.
- In the Start menu or Finder of your local machine, you should find Git Bash. Open that agent.
- The result should look familiar. You are at a command line prompt. Rather than being on the class shared server as you were in section 1.7 Command Line, however, you are on your local machine.

Error Correction

- If anything goes wrong with this lesson and you’re aren’t sure what to do or you can’t get out of something, close your command line window.
- Then, reopen Git Bash.
- You may lose some work, but once you figure out what state the project is in (git status), you’ll be able to continue.

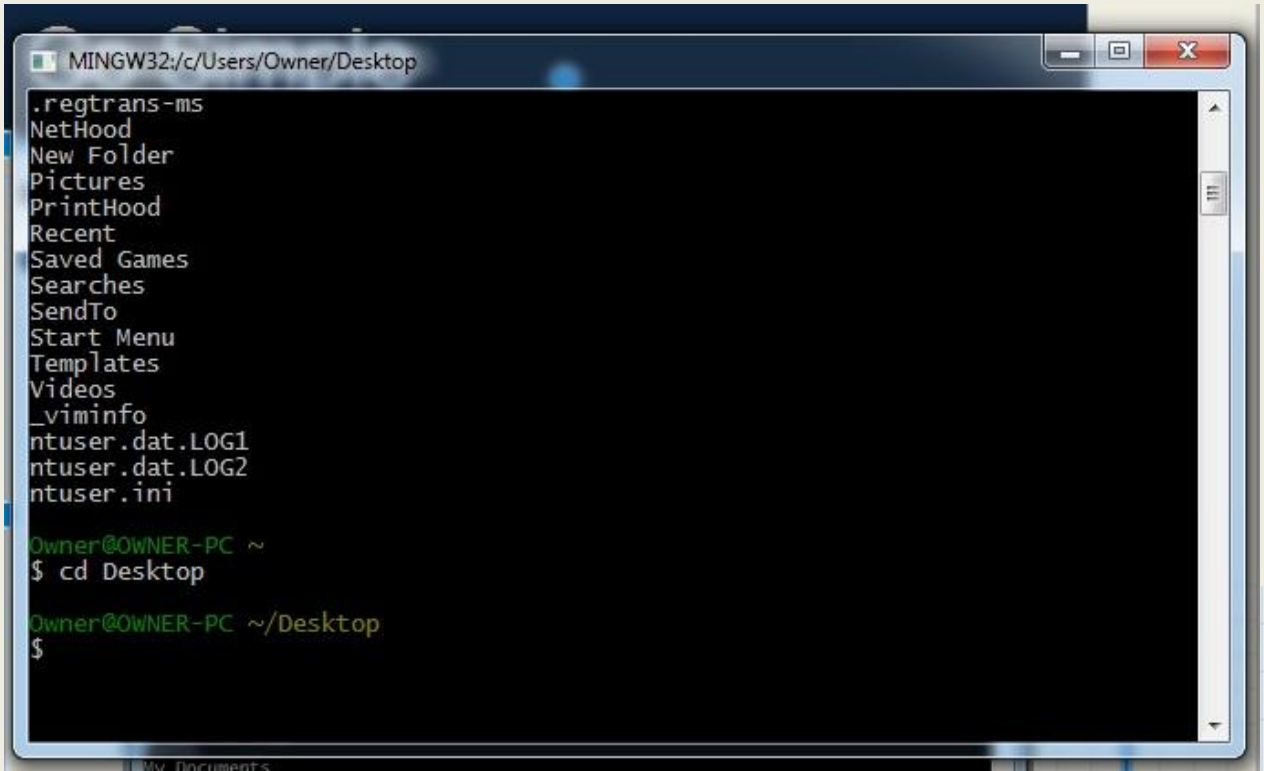
Let’s Review

- Type “pwd” to see where you are.
- Type “ls” to see what’s here. Could be a long list. You might have to scroll up.



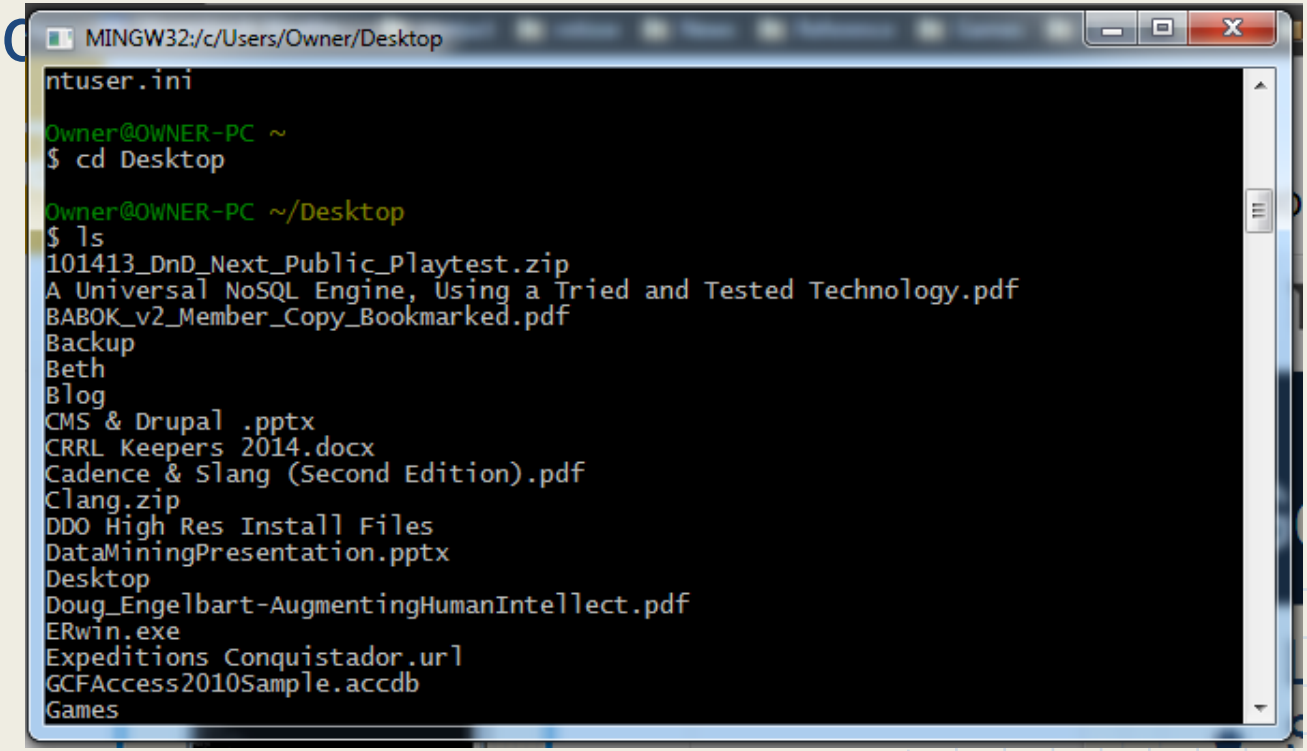
Go Simple

- Let’s move to the Desktop.
- Scroll back to the prompt (if needed) and type “cd Desktop”.



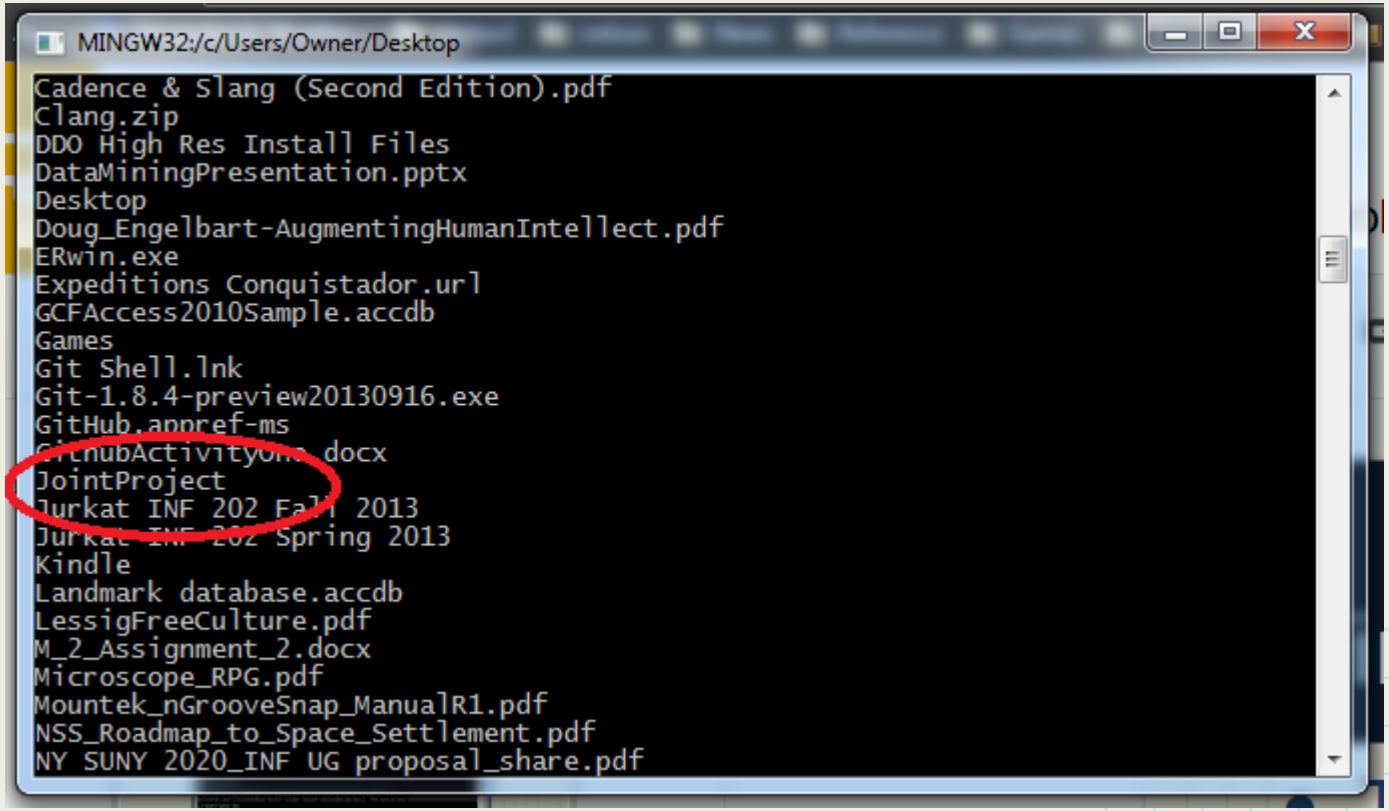
Find Folder

- Type “ls” to see what’s here.
- Could be lots of files (depending on how crowded your desktop is). Scrolling may be need



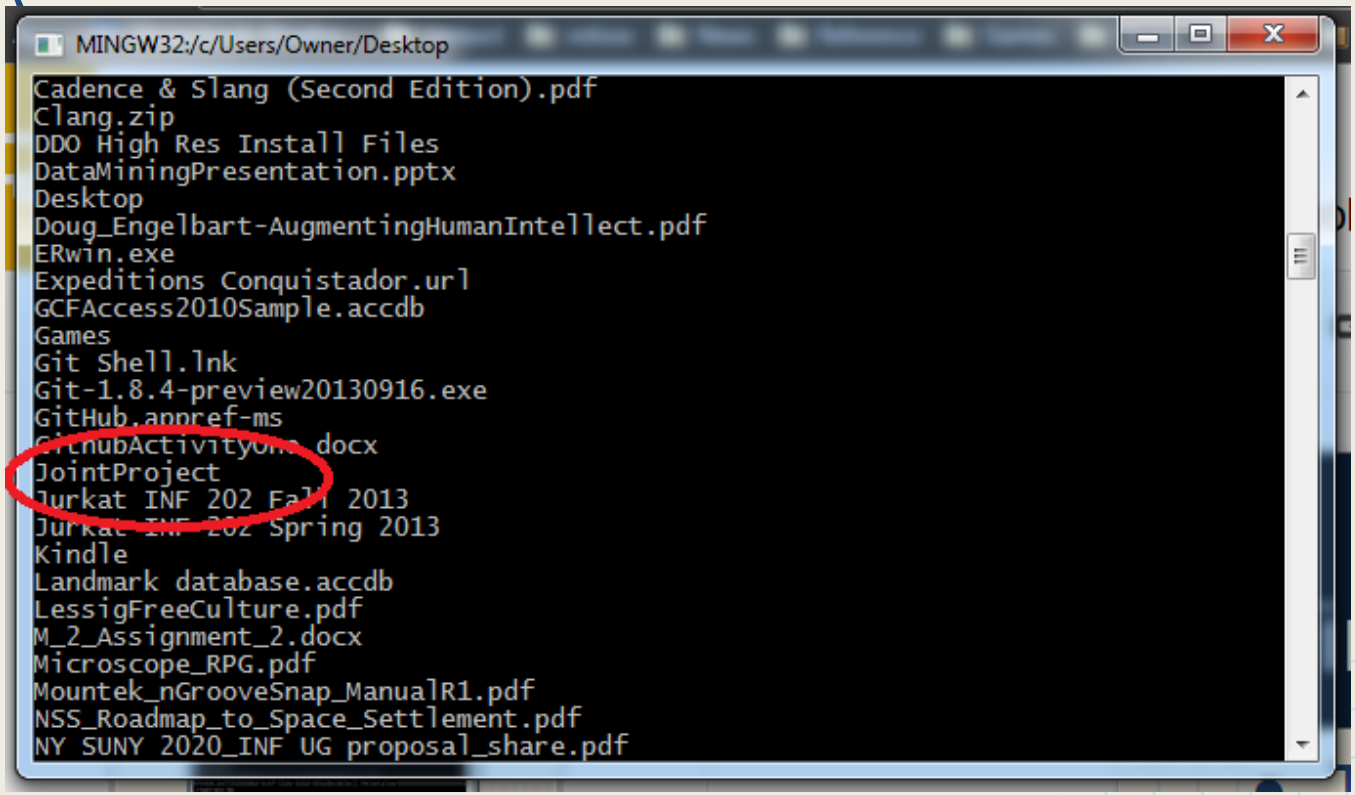
Find Folder

- Look for the “JointProject” folder we created in Stage 1.



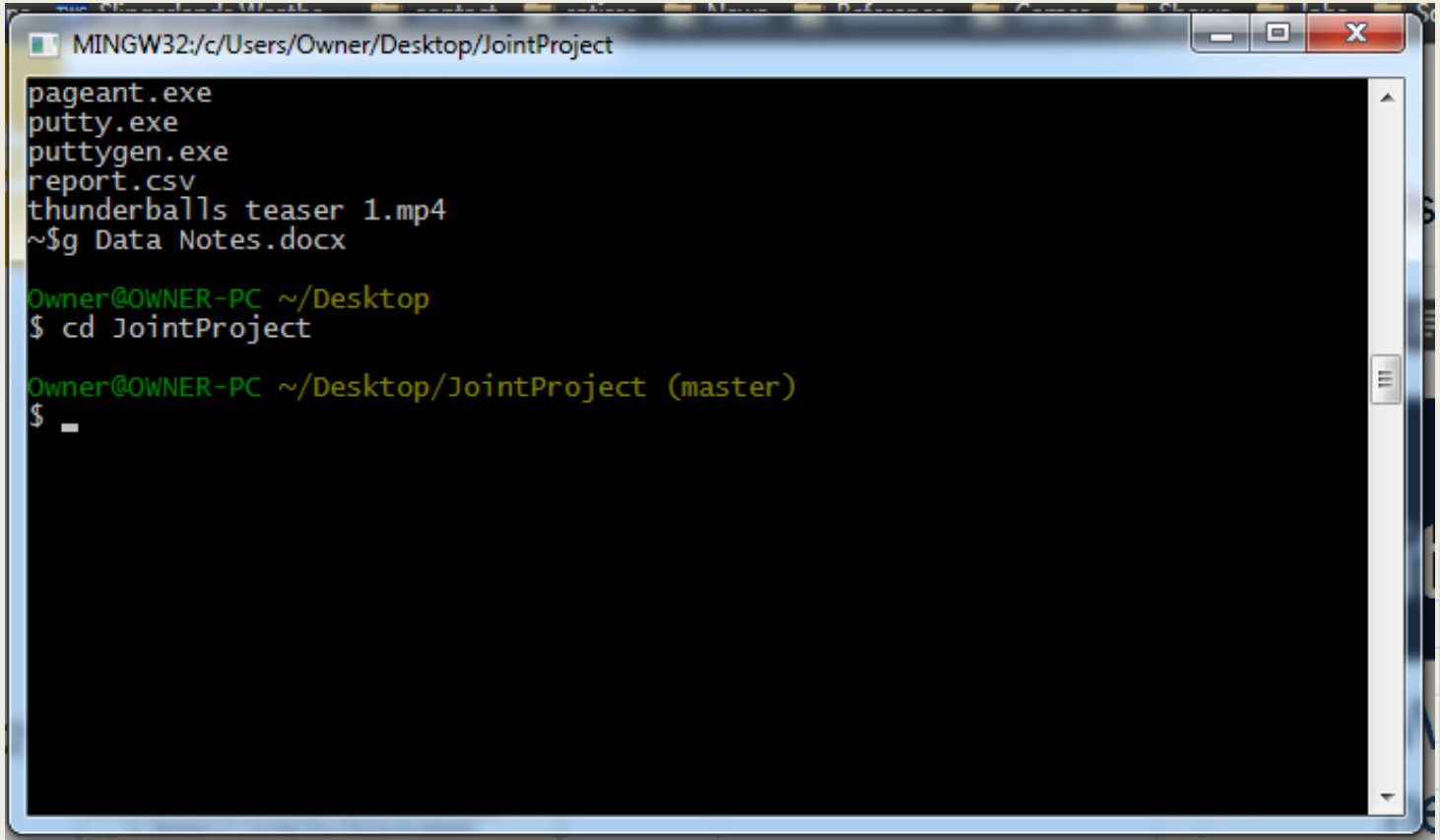
Find Folder

- If it's not there, search around your system for it or create it again (redo Stage 1, slides 19-23)



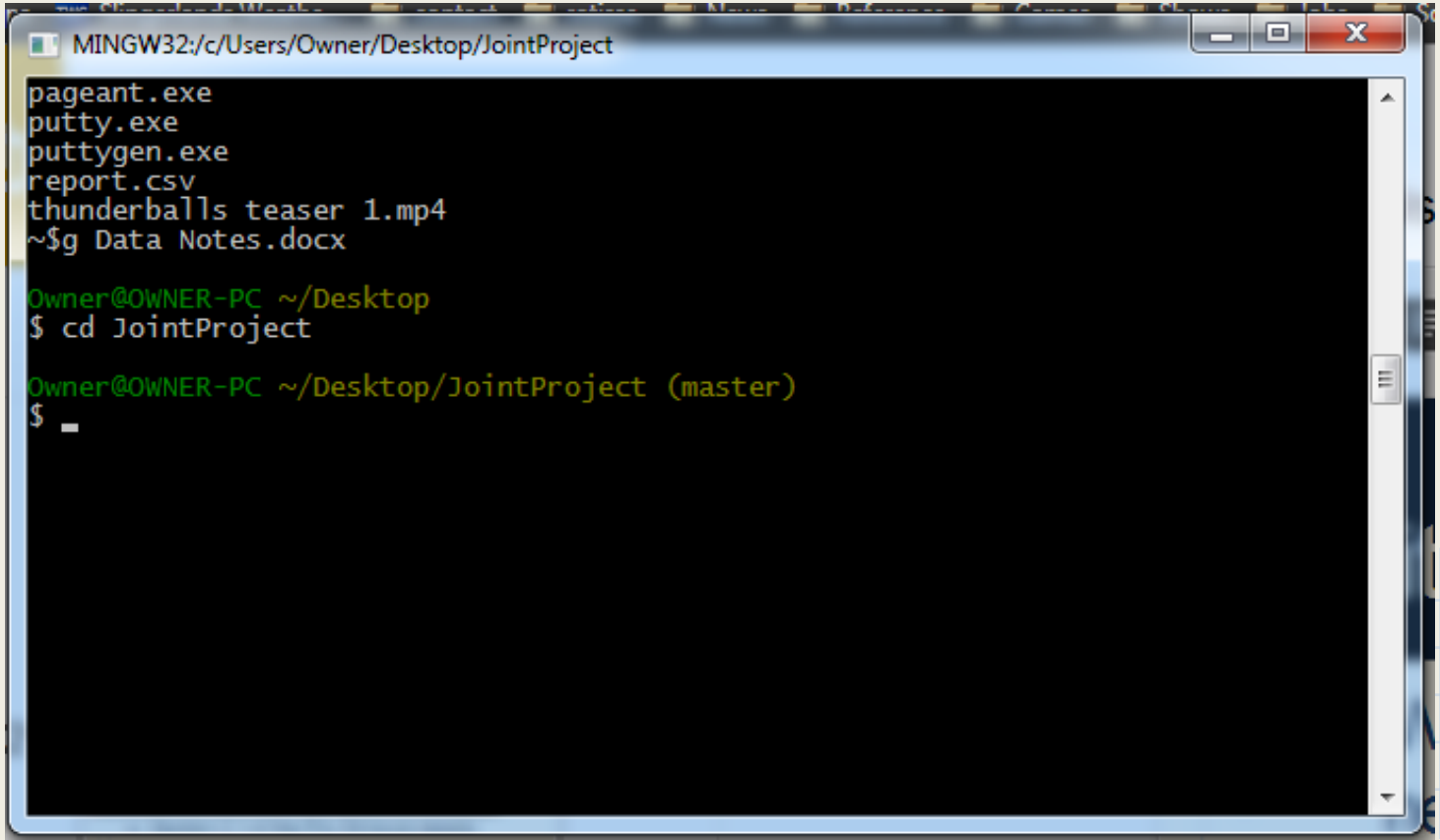
Move into JointProject

- Enter the JointProject folder by typing “cd JointProject”.



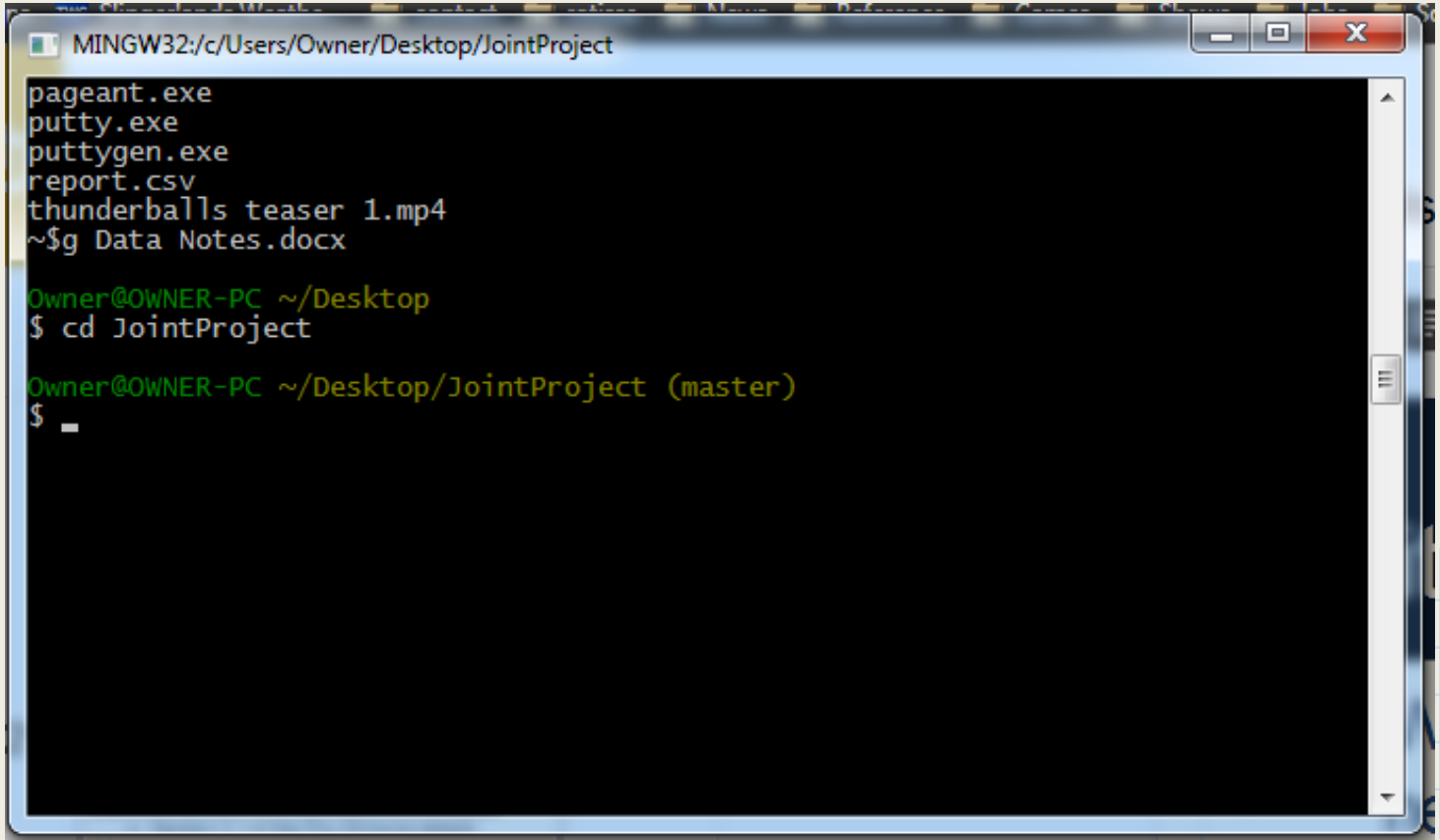
No Need for Git Init

- The “(master)” designation tells you (1) Git is operating (it was initiated in Stage 1) and . . .

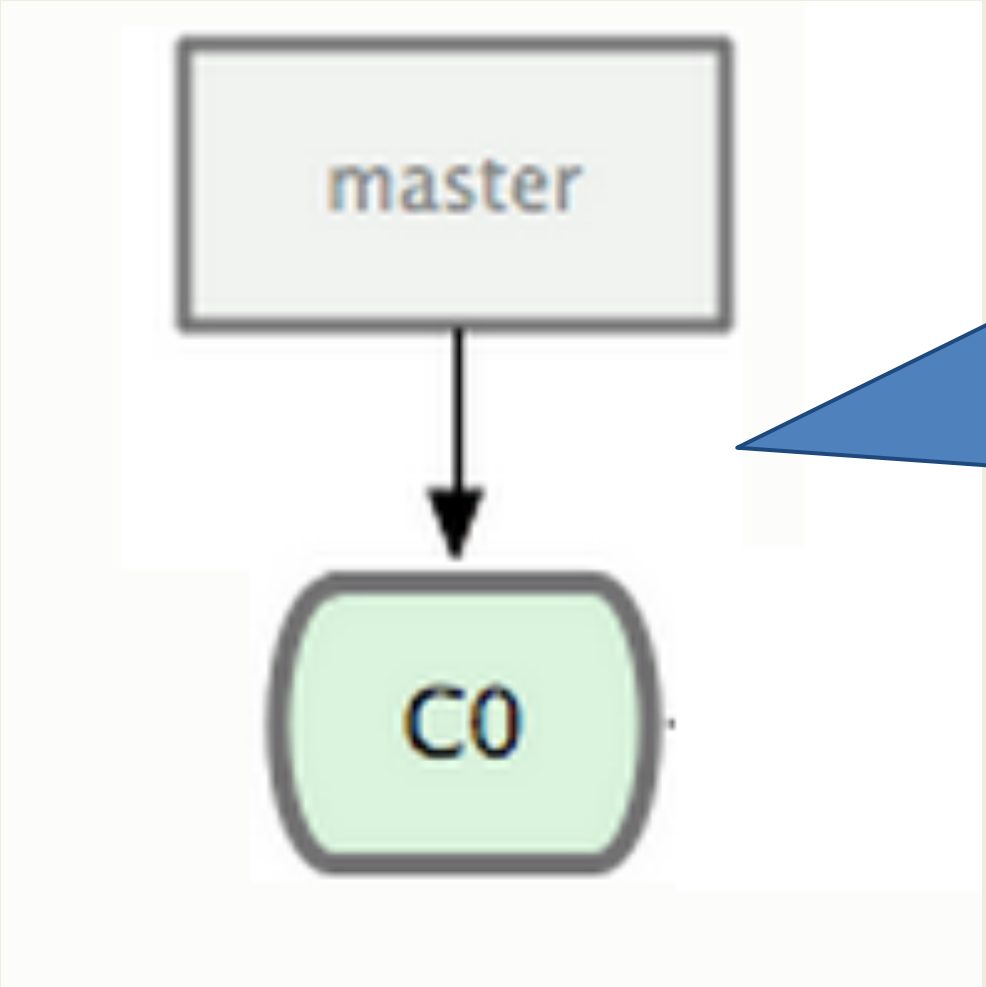


Look Ma, It's a Branch

- (2) you are on the master branch -- the main snapshot of your project files.



C0 Project State



Assume the initial state of your JointProject work is called C0. The master branch is a pointer to the latest commit snapshot of your project.

Creating a Project History

- In the next several slides, you'll make some changes to the README file and commit them (with a comment).
- That new project state can be called C1.
- You'll also create a new file READMETOO and commit it (with a comment).
- At that point (slide 31), the project state can be called C2.
- By that time, you will have a project history.

Making Changes

MINGW32:/c/Users/Owner/Desktop/JointProject

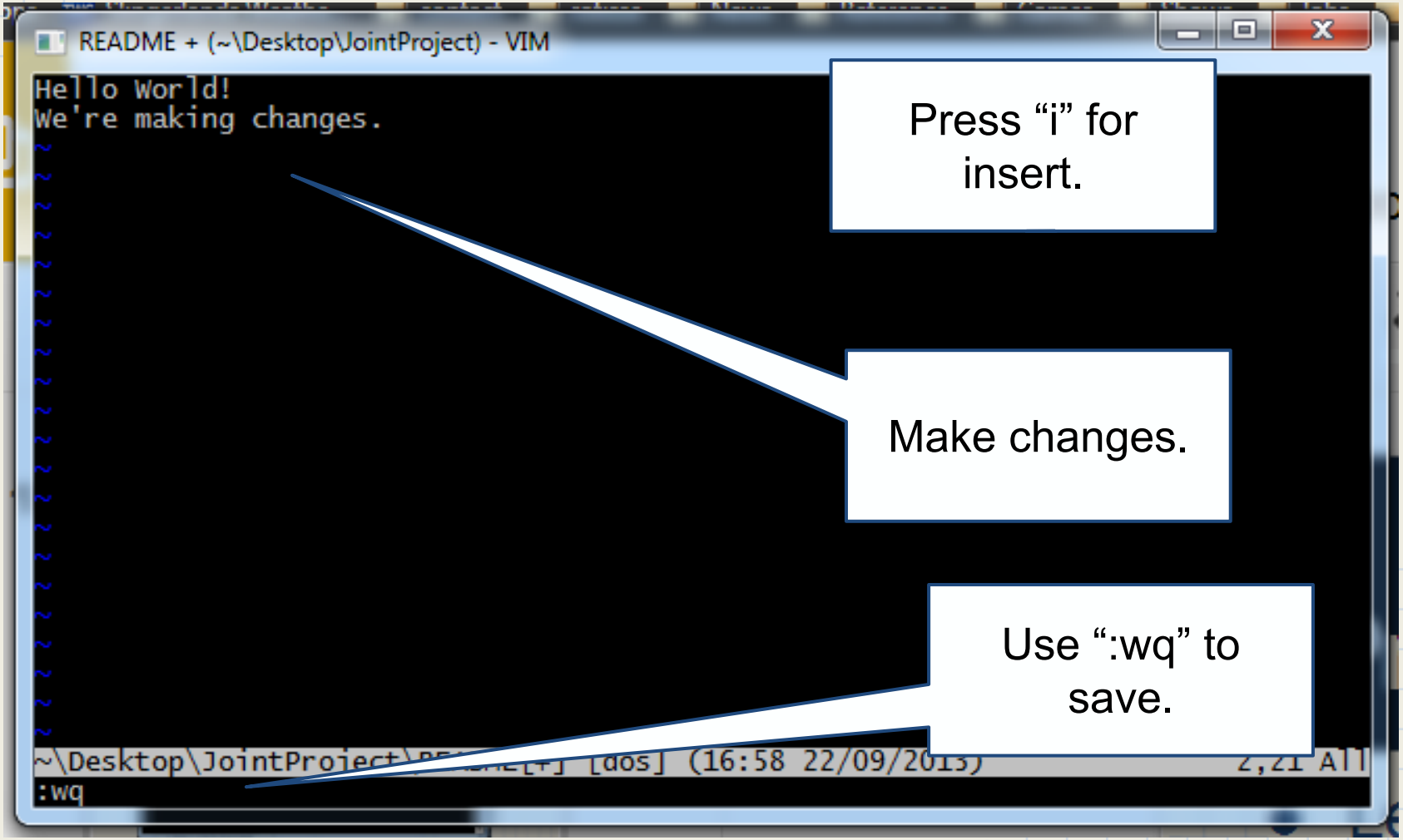
```
pageant.exe
putty.exe
puttygen.exe
report.csv
thunderballs teaser 1.mp4
~$g Data Notes.docx

Owner@OWNER-PC ~/Desktop
$ cd JointProject

Owner@OWNER-PC ~/Desktop/JointProject (master)
$ ls
README

Owner@OWNER-PC ~/Desktop/JointProject (master)
$ git status
# On branch master
nothing to commit, working directory clean

Owner@OWNER-PC ~/Desktop/JointProject (master)
$ vim README
```

What's in this folder?

What's the git status?

Let's edit the README file.

Vim for the Change



Staging and Status

MINGW32:/c/Users/Owner/Desktop/JointProject

```
Owner@OWNER-PC ~/Desktop/JointProject (master)
$ git status
# On branch master
# Changes not staged for commit:
#   (use "git add <file>..." to update what will be committed)
#   (use "git checkout -- <file>..." to discard changes in working directory)
#
#       modified:   README
#
no changes added to commit (use "git add" and/or "git commit -a")

Owner@OWNER-PC ~/Desktop/JointProject (master)
$ git add README

Owner@OWNER-PC ~/Desktop/JointProject (master)
$ git status
# On branch master
# Changes to be committed:
#   (use "git reset HEAD <file>..." to unstage)
#
#       modified:   README
#

Owner@OWNER-PC ~/Desktop/JointProject (master)
$ _
```

"Git status" to see untracked (not staged) changes.

"Git add" to stage the changes.

"Git status" to see the staged changes.

Commit to Create New Snapshot (C1)

MINGW32:/c/Users/Owner/Desktop/JointProject

```
Owner@OWNER-PC ~/Desktop/JointProject (master)
$ git commit
```

"Git commit" opens Vim editor.

COMMIT_EDITMSG + (~\Desktop\JointProject\.git) - VIM

```
Changes that create version C1 of project.
# Please enter the commit message for your changes. Lines starting
# with '#' will be ignored, and an empty message aborts the commit.
# On branch master
# Changes to be committed:
#   (use "git reset HEAD <file>..." to unstage)
#
#       modified:   README
#
~
~
~
~
~
~
~
~
~
~
~
~\Desktop\JointProject\.git\COMMIT_EDITMSG[+] [unix] (22:59 24/10/2013) 1,42 All
:wq
```

Use Vim command "i" and enter comment.

":wq" to save comment.

Commit Feedback and Log

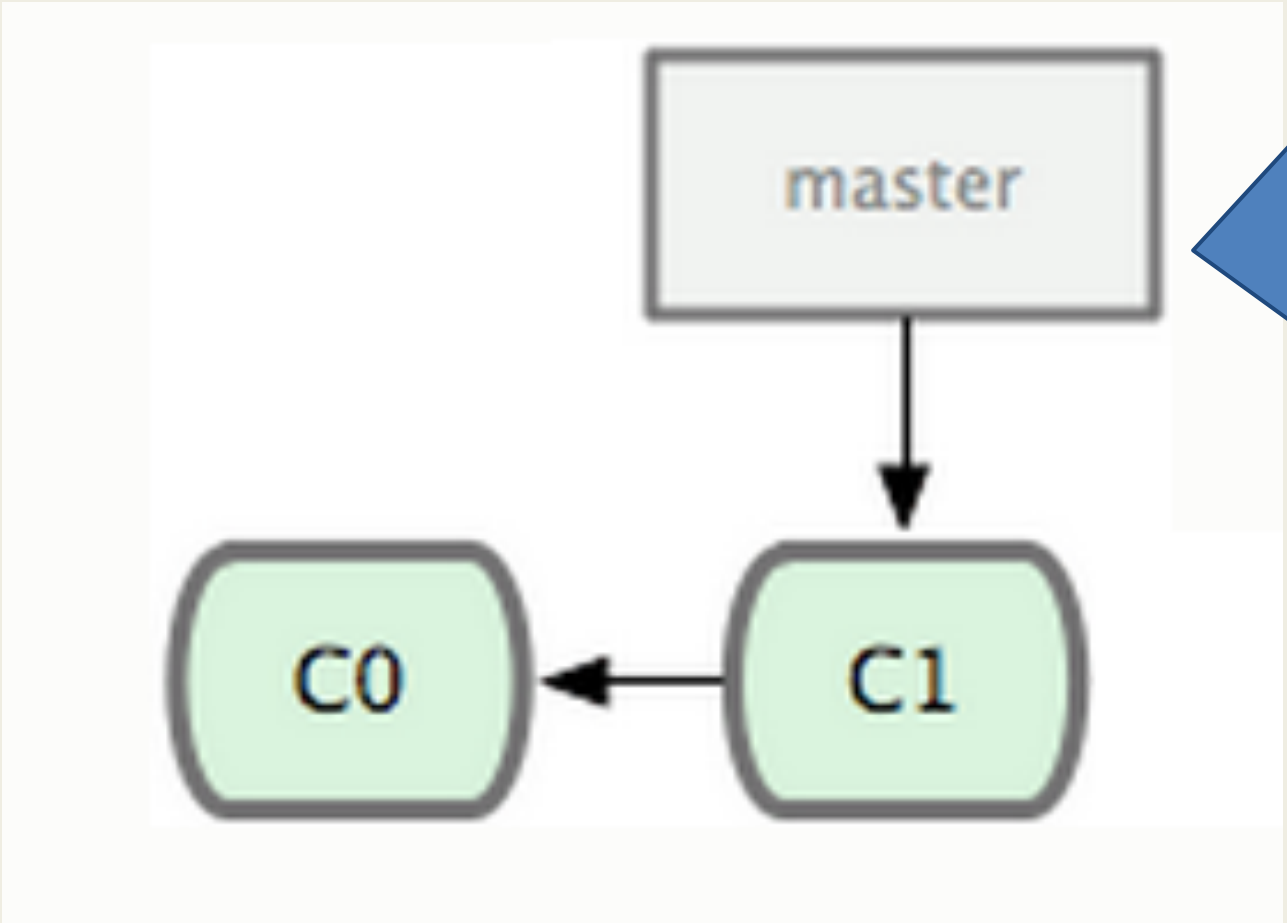
MINGW32:/c/Users/Owner/Desktop/JointProject

```
Owner@OWNER-PC ~/Desktop/JointProject (master)
$ git commit
[master fbd9728] Changes that create version C1 of project.
 1 file changed, 1 insertion(+)

Owner@OWNER-PC ~/Desktop/JointProject (master)
$ git log
commit fbd97283f4536cf2e8756e8a02f495e44704c334
Author: M Alexander Jurkat <alex.jurkat@gmail.com>
Date:   Thu Oct 24 22:59:39 2013 -0400

    Changes that create version C1 of project.

commit 3066da9b5c37ba34708b4922618948ae80319218
Author: M Alexander Jurkat <alex.jurkat@gmail.com>
Date:   Sun Sep 22 20:29:54 2013 -0400

    My first commit!

Owner@OWNER-PC ~/Desktop/JointProject (master)
$ _
```

Commit feedback summarizes changes.

"Git log" shows two snapshots. The first you did during Stage 1; the second you did just now.

C1 Project State



With the new commit, the master branch pointer moves automatically to the new snapshot of your project (C1).

Changes for C2

MINGW32:/c/Users/Owner/Desktop/JointProject

```
Owner@OWNER-PC ~/Desktop/JointProject (master)
$ git status
# On branch master
nothing to commit, working directory clean

Owner@OWNER-PC ~/Desktop/JointProject (master)
$ vim README

Owner@OWNER-PC ~/Desktop/JointProject (master)
$ vim READMETOO

Owner@OWNER-PC ~/Desktop/JointProject (master)
$ git status
# On branch master
# Changes not staged for commit:
#   (use "git add <file>..." to update what will be committed)
#   (use "git checkout -- <file>..." to discard changes in working directory)
#
#       modified:   README
#
# Untracked files:
#   (use "git add <file>..." to include in what
#
#       READMETOO
no changes added to commit (use "git add" and/or "git co

Owner@OWNER-PC ~/Desktop/JointProject (master)
$
```

"Git status" -- you're on master branch.

Vim to make more changes to README (editor screen not shown).

Vim to create new file READMETOO (editor screen not shown).

"Git status" to see the unstaged/untracked changes. Old, modified file unstaged; new file untracked.

Committing for C2

MINGW32:/c/Users/Owner/Desktop/JointProject

```
Owner@OWNER-PC ~/Desktop/JointProject (master)
$ git add README

Owner@OWNER-PC ~/Desktop/JointProject (master)
$ git add READMETOO

Owner@OWNER-PC ~/Desktop/JointProject (master)
$ git status
# On branch master
# Changes to be committed:
#   (use "git reset HEAD <file>..." to unstage)
#
#       modified:   README
#       new file:   READMETOO
#

Owner@OWNER-PC ~/Desktop/JointProject (master)
$ git commit
```

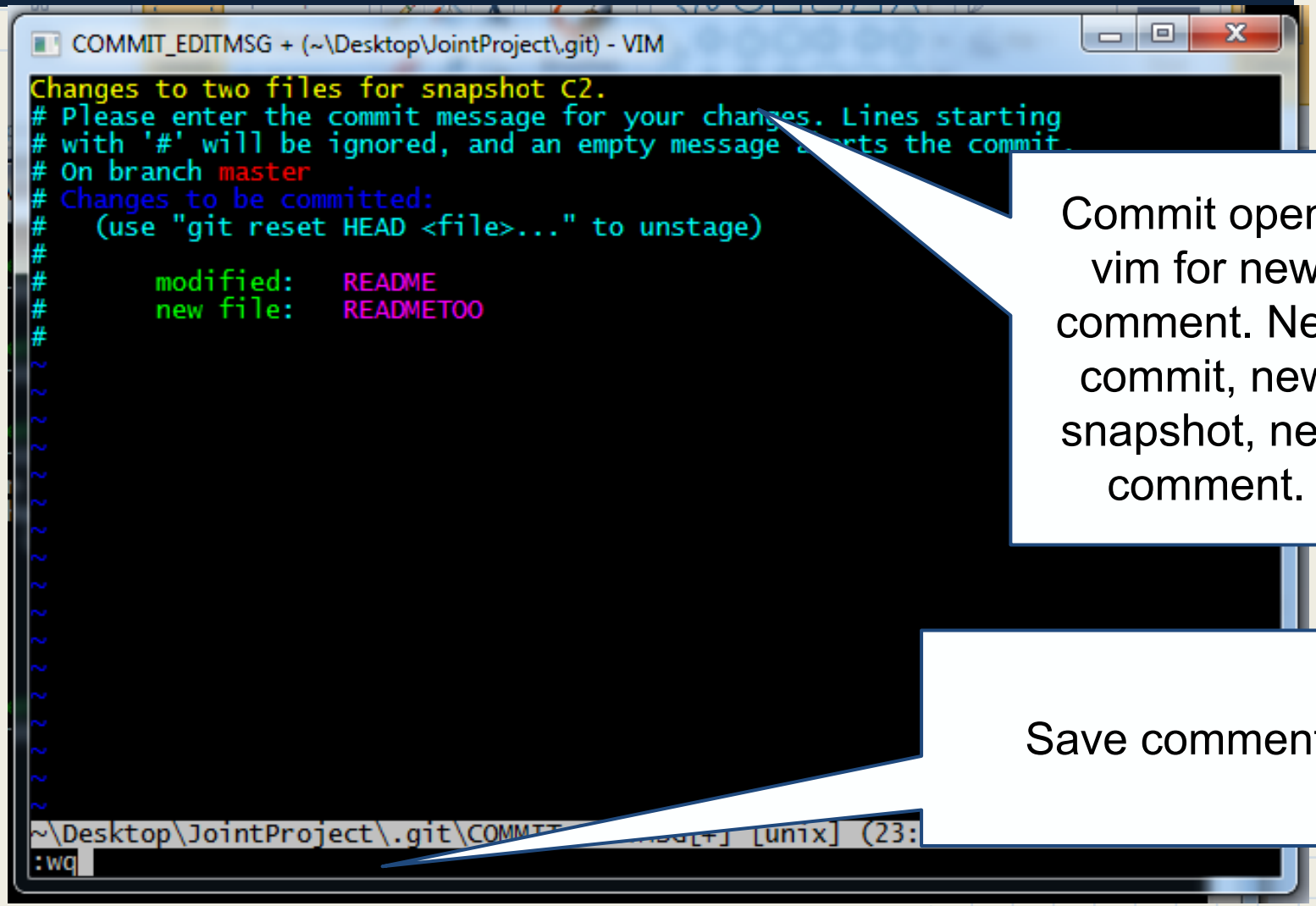
README now staged.

READMETOO now staged.

Confirm that files are staged.

Time to commit.

Commenting for C2



Commit opens vim for new comment. New commit, new snapshot, new comment.

Save comment.

C2 Log and Status

MINGW32:/c/Users/Owner/Desktop/JointProject

```
Owner@OWNER-PC ~/Desktop/JointProject (master)
$ git commit
[master 6bc4c06] Changes to two files for snapshot C2.
 2 files changed, 2 insertions(+)
 create mode 100644 READMETOO

Owner@OWNER-PC ~/Desktop/JointProject (master)
$ git log
commit 6bc4c06e855c4e62c1cc70110182142bf162e748
Author: M Alexander Jurkat <alex.jurkat@gmail.com>
Date:   Thu Oct 24 23:32:57 2013 -0400

    Changes to two files for snapshot C2.

commit fbd97283f4536cf2e8756e8a02f495e44704c334
Author: M Alexander Jurkat <alex.jurkat@gmail.com>
Date:   Thu Oct 24 22:59:39 2013 -0400

    Changes that create version C1 of project.

commit 3066da9b5c37ba34708b4922618948ae80319218
Author: M Alexander Jurkat <alex.jurkat@gmail.com>
Date:   Sun Sep 22 20:29:54 2013 -0400

    My first commit!

Owner@OWNER-PC ~/Desktop/JointProject (master)
$ git status
# On branch master
nothing to commit, working directory clean

Owner@OWNER-PC ~/Desktop/JointProject (master)
$ _
```

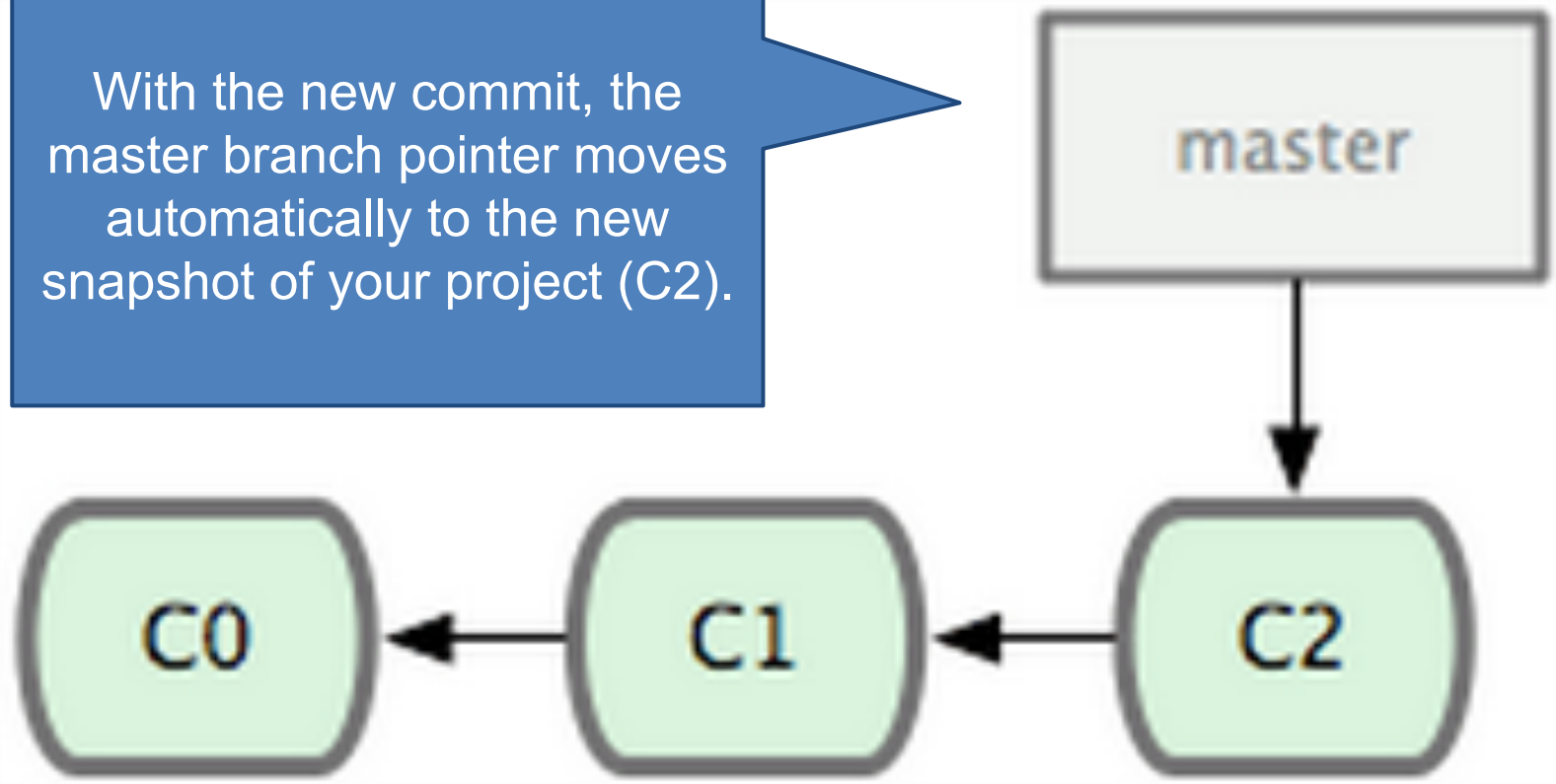
Commit feedback.

“Git log” shows development of project from C0 (when you started) to C2 (most recent commit).

“Git status” shows you’re still on master branch.

C2 Project State

With the new commit, the master branch pointer moves automatically to the new snapshot of your project (C2).



Commit Pro

Time to
SMILE!

Multi-Branch

- Let’s say you want to experiment with your JointProject work. (Not terribly meaningful when the project consists of a twice modified README file and a new READMETOO file, but work with me here.)
- You don’t want to lose the work you’ve done so far on JointProject, so you create a new branch called “experiment”.

Create Experiment

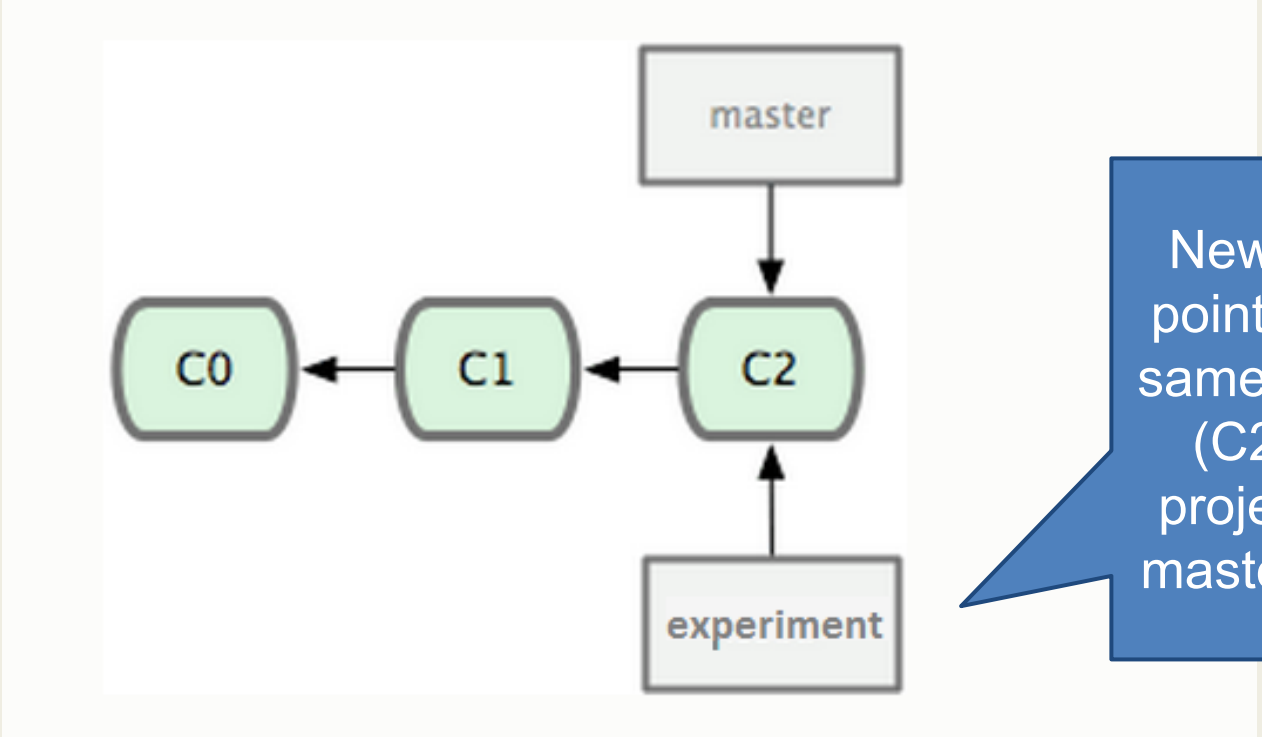
MINGW32:/c/Users/Owner/desktop/JointProject

```
Owner@OWNER-PC ~/desktop/JointProject (master)
$ git branch experiment

Owner@OWNER-PC ~/desktop/JointProject (master)
$ git checkout experiment
Switched to branch 'experiment'

Owner@OWNER-PC ~/desktop/JointProject (experiment)
$
```

"Git branch" to create new branch

"Git checkout" to move to that new branch.

Create Experiment



New branch, pointing to the same snapshot (C2) of the project as the master branch.

Experimenting

- Now you can make whatever changes you like to the experiment branch and the master branch version of the project remains intact.
- Open the READMETOO file using Vim.
- Add a new line “Experimenting away on the thin ice of the new day.” (reference, anyone?)
- Save, stage, and commit the modification.
- As newly designated Commit Pros, no step-by-step guide needed. You can do this!

Experimenting Summa

MINGW32:/c/Users/Owner/desktop/JointProject

```
Owner@OWNER-PC ~/desktop/JointProject (master)
$ git branch experiment

Owner@OWNER-PC ~/desktop/JointProject (master)
$ git checkout experiment
Switched to branch 'experiment'

Owner@OWNER-PC ~/desktop/JointProject (experiment)
$ vim READMETOO

Owner@OWNER-PC ~/desktop/JointProject (experiment)
$ git add READMETOO

Owner@OWNER-PC ~/desktop/JointProject (experiment)
$ git commit
[experiment 958c085] Experimenting with READMETOO, creating C3
 1 file changed, 1 insertion(+)

Owner@OWNER-PC ~/desktop/JointProject (experiment)
$ git log
commit 958c085f28bc4219e1fb969e2feb4b8d0e0d2ac3
Author: M Alexander Jurkat <alex.jurkat@gmail.com>
Date:   Sat Oct 26 12:04:36 2013 -0400

    Experimenting with READMETOO, creating C3

commit 6bc4c06e855c4e62c1cc70110182142bf162e748
Author: M Alexander Jurkat <alex.jurkat@gmail.com>
Date:   Thu Oct 24 23:32:57 2013 -0400
```

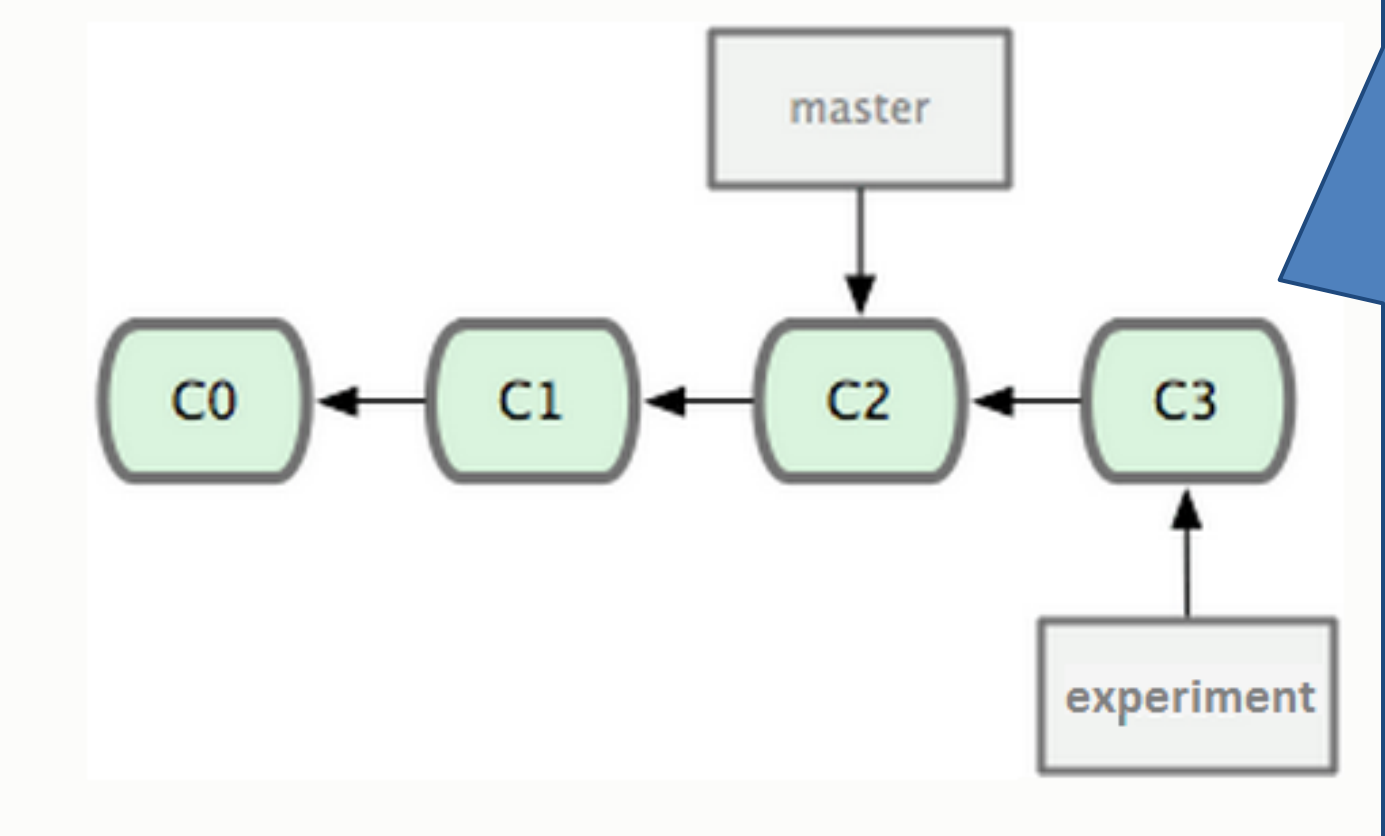
READMETOO modified (editor screen not shown)

Modified READMETOO staged.

Modified READMETOO committed (comment editor screen not shown).

“Git log” shows new commit comment (prior comments not all shown).

JointProject Snapshot C3



Experiment branch automatically moves to snapshot C3 which was created by the new commit. Master branch stays pointing to prior version (C2) of JointProject.

Doubting Yourself

- You’re not sure about the reference in “Experimenting away on the thin ice of the new day.” You’re thinking it might be too old school.
- You’re thinking of trying something else, but you don’t want to lose your work on the experiment branch.
- No problem. We’ve still got the master branch preserved.

New Branch

- Time to head back to the master branch. Use “git checkout master”.
- Then start a new branch for an alternative approach to experiment. Call the new branch “surething” (“git branch surething”).
- Move to the new branch (“git checkout surething”).
- Open READMETOO using vim.

Surething

MINGW32:/c/Users/Owner/desktop/JointProject

```
Owner@OWNER-PC ~/desktop/JointProject (experiment)
$ git checkout master
Switched to branch 'master'

Owner@OWNER-PC ~/desktop/JointProject (master)
$ git branch surething

Owner@OWNER-PC ~/desktop/JointProject (master)
$ git checkout surething
Switched to branch 'surething'

Owner@OWNER-PC ~/desktop/JointProject (surething)
$ vim READMETOO
```

Back to master branch.

Create surething branch.

Move to surething branch

Open READMETOO

Working on Surething

- Once you open READMETOO, you'll notice that the "Experimenting . . ." line is not there. Remember we're working off of snapshot C2.
- Add a new line: "Experimenting gangnam style." You're pretty sure that reference can't be missed.
- Save, stage, and commit the new READMETOO file.
- Check the log.

Committing Surething

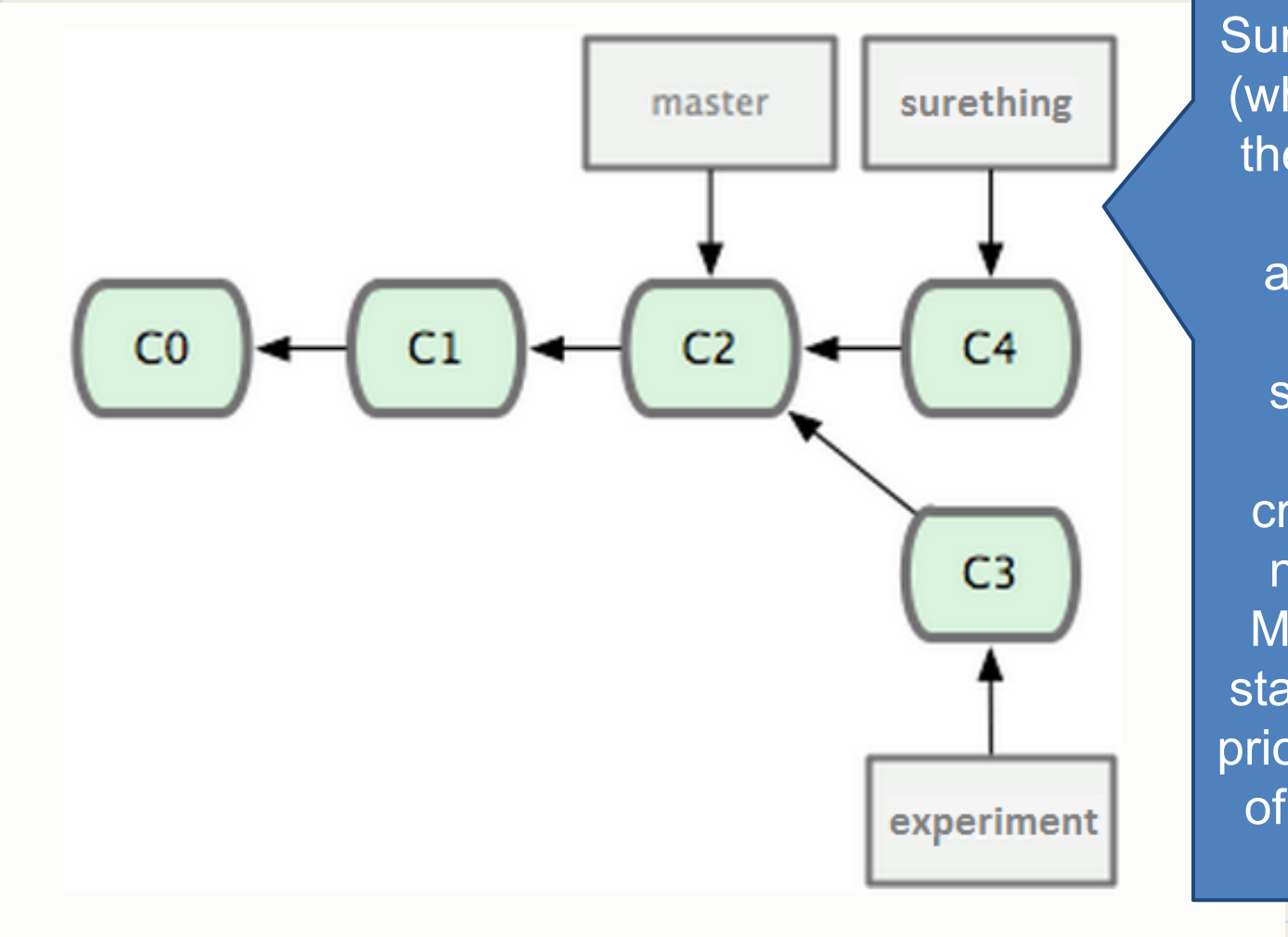
Stage the modified READMETOO file

```
MINGW32:/c/Users/Owner/desktop/JointProject

Owner@OWNER-PC ~/desktop/JointProject (surething)
$ git add READMETOO

Owner@OWNER-PC ~/desktop/JointProject (surething)
$ git commit READMETOO
[surething 560f052] Can't miss reference in snapshot C4.
 1 file changed, 1 insertion(+)

Owner@OWNER-PC ~/desktop/JointProject (surething)
$ git log
commit 560f05269f4660e358890a12d6e555b09c267569
Author: M Alexander Jurkat <alex.jurkat@gmail.com>
Date:   Sat Oct 26 12:37:56 2013 -0400

    Can't miss reference in snapshot C4.

commit 6bc4c06e855c4e62c1cc70110182142bf162e748
Author: M Alexander Jurkat <alex.jurkat@gmail.com>
Date:   Thu Oct 24 23:32:57 2013 -0400

    Changes to two files for snapshot C2.
```

Commit the changes (“READMETOO” portion of command not needed -- see slide 37).

“Git log” shows commits and comments (full log not shown). Notice that comment about snapshot C3 doesn’t exist. That snapshot is part of the experiment branch, not this one (see slide 37).

JointProject Snapshot C4



Surething branch (which started in the same place as master) automatically moves to snapshot C4 which was created by the new commit. Master branch stays pointing to prior version (C2) of JointProject.

Feeling Good about Surething

- The more you think about your changes in surething, the better you feel about them.
- You're ready to bring the master branch of the project to the same state as surething.
- That done by merging the two snapshots.
- First, switch back to master.
- Next, merge the two snapshots using "git merge".
- Last, clean up a bit by deleting surething.

Merging Master and Suret

Move from surething branch to master branch.

MINGW32:/c/Users/Owner/desktop/JointProject

```
Owner@OWNER-PC ~/desktop/JointProject (surething)
$ git checkout master
Switched to branch 'master'

Owner@OWNER-PC ~/desktop/JointProject (master)
$ git merge surething
Updating 6bc4c06..560f052
Fast-forward
 READMETOO | 1 +
 1 file changed, 1 insertion(+)

Owner@OWNER-PC ~/desktop/JointProject (master)
$ git branch -d surething
Deleted branch surething (was 560f052).

Owner@OWNER-PC ~/desktop/JointProject (master)
$ _
```

Merge surething into master. “Fast-forward” is announced because you are bringing the master branch forward. Master is a direct ancestor of surething (no divergent branch to account for).

Surething goes bye-bye.

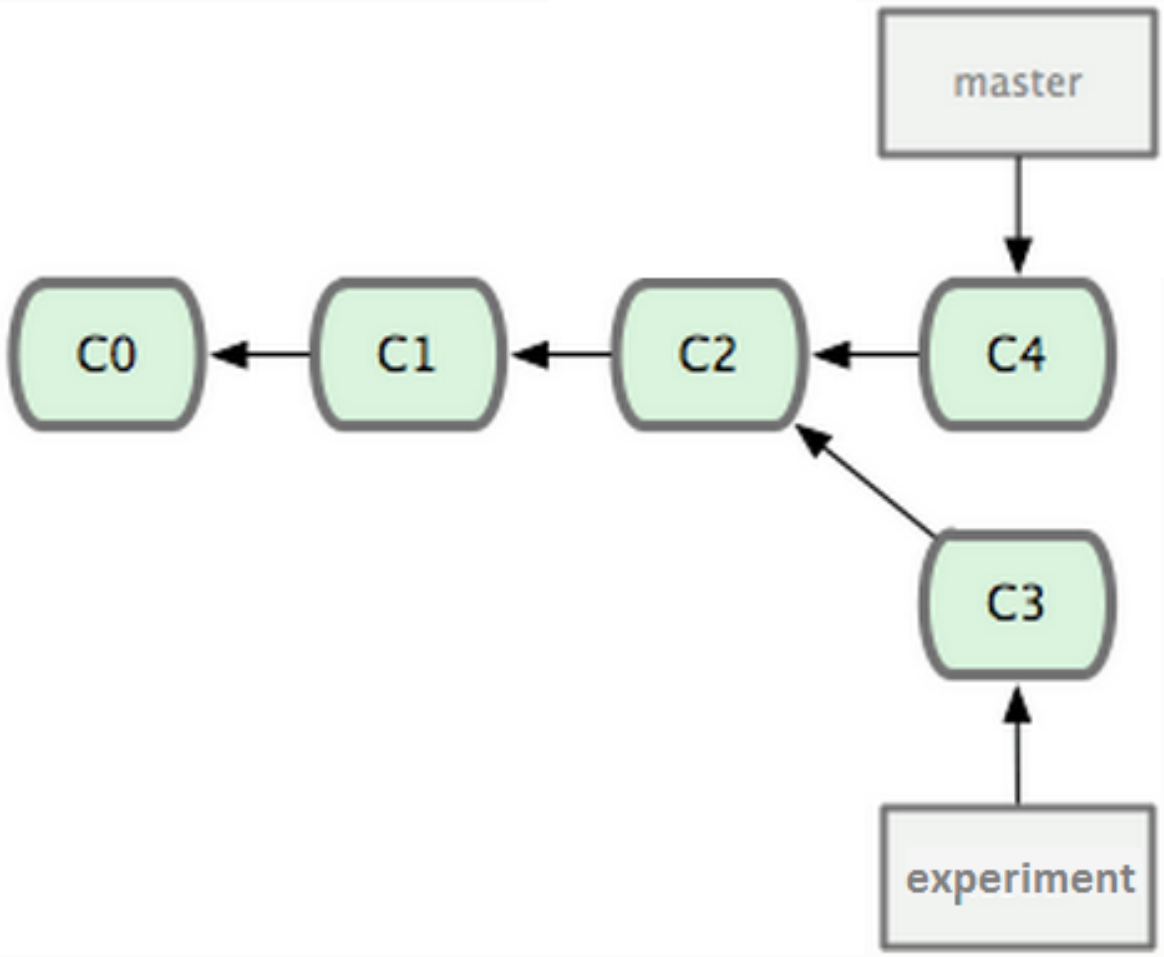
Merging Master and Surething

MINGW32:/c/Users/Owner/desktop/JointProject

```
Owner@OWNER-PC ~/desktop/JointProject (master)
$ git log
commit 560f05269f4660e358890a12d6e555b09c267569
Author: M Alexander Jurkat <alex.jurkat@gmail.com>
Date:   Sat Oct 26 12:37:56 2013 -0400

    Can't miss reference in snapshot C4.

commit 6bc4c06e855c4e62c1cc70110182142bf162e748
Author: M Alexander Jurkat <alex.jurkat@gmail.com>
Date:   Thu Oct 24 23:32:57 2013 -0400

    Changes to two files for snapshot C2.

commit fbd97283f4536cf2e8756e8a02f495e44704c334
Author: M Alexander Jurkat <alex.jurkat@gmail.com>
Date:   Thu Oct 24 22:59:39 2013 -0400

    Changes that create version C1 of project.

commit 3066da9b5c37ba34708b4922618948ae80319218
Author: M Alexander Jurkat <alex.jurkat@gmail.com>
Date:   Sun Sep 22 20:29:54 2013 -0400

    My first commit!

Owner@OWNER-PC ~/desktop/JointProject (master)
$ _
```

"Git log" of newly merged master branch looks the same as the log of the former surething branch. You essentially renamed surething as master and made it the main snapshot of the project.

JointProject Snapshot C4



Master branch has taken the place of the former surething branch and now points to snapshot C4.

New Thoughts on Experiment

- Suddenly, you have a brainstorm about experiment.
- You've decided to add a hint about the obscure reference.
- You move to the experiment branch.
- You open READMETOO and add "War Child 1974." to the file.
- Save, stage, and commit.
- Check the log.

Experimenting Anew

Move from master to experiment branch.

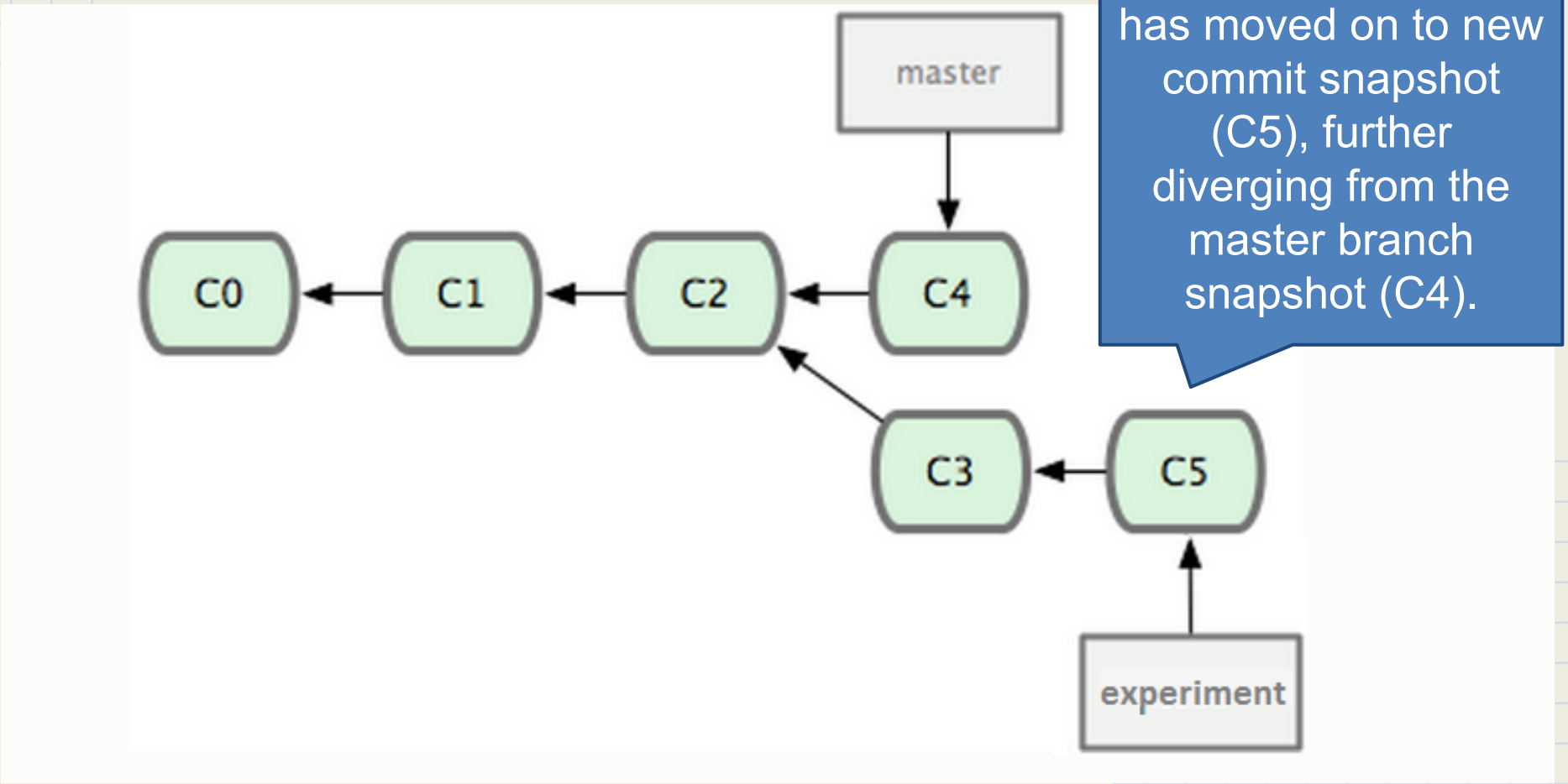
MINGW32:/c/Users/Owner/desktop/JointProject

```
Owner@OWNER-PC ~/desktop/JointProject (master)
$ git checkout experiment
Switched to branch 'experiment'

Owner@OWNER-PC ~/desktop/JointProject (experiment)
$ vim READMETOO

Owner@OWNER-PC ~/desktop/JointProject (experiment)
$ git add READMETOO

Owner@OWNER-PC ~/desktop/JointProject (experiment)
$ git commit
[experiment b9cd67b] Adding hint to obscure reference, creating sna
 1 file changed, 1 insertion(+)

Owner@OWNER-PC ~/desktop/JointProject (experiment)
$ git log
commit b9cd67bc38ebf44ff60a8e128929ff49094c95f5
Author: M Alexander Jurkat <alex.jurkat@gmail.com>
Date:   Sat Oct 26 14:31:58 2013 -0400

    Adding hint to obscure reference, creating snapshot C5.

commit 958c085f28bc4219e1fb969e2feb4b8d0e0d2ac3
Author: M Alexander Jurkat <alex.jurkat@gmail.com>
Date:   Sat Oct 26 12:04:36 2013 -0400

    Experimenting with READMETOO, creating C3

commit 6bc4c06e855c4e62c1cc70110182142bf162e748
Author: M Alexander Jurkat <alex.jurkat@gmail.com>
Date:   Thu Oct 24 23:32:57 2013 -0400

    Changes to two files for snapshot C2.
```

Editing READMETOO. Staging READMETOO. Committing changes (only changes were to READMETOO so no need to reference that file).

Log now shows new commit (C5). You'll notice no comment mentions snapshot C4. That was part of surething, and has now been merged into master. C4 has never been part of the experiment branch.

JointProject Snapshot C5

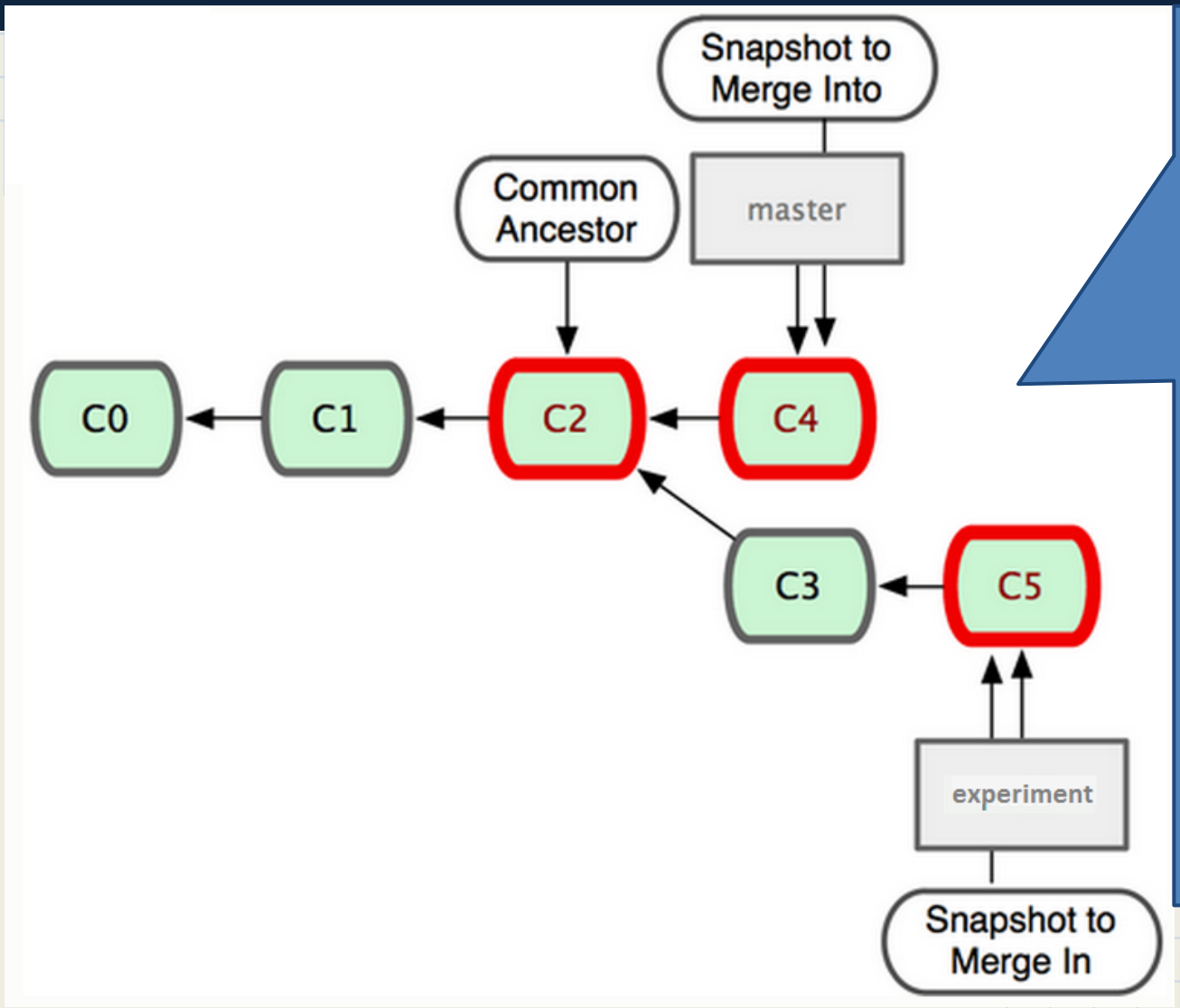


Experiment branch has moved on to new commit snapshot (C5), further diverging from the master branch snapshot (C4).

Bringing It All Home

- You’re now comfortable with the state of experiment. It’s time to bring your work together, all in one place.
- Time to switch back to the master branch.
- Then merge the branches using “git merge experiment”.

JointProject C4-C5 Merge Plan



Master is NOT a direct ancestor of experiment, so Git can't just fast forward the master branch to a new merged project snapshot. Git finds the closes common ancestor of the two state and brings all three together.

Houston, We Have A Problem

MINGW32:/c/Users/Owner/desktop/JointProject

```
Owner@OWNER-PC ~/desktop/JointProject (experiment)
$ git checkout master
Switched to branch 'master'

Owner@OWNER-PC ~/desktop/JointProject (master)
$ git merge experiment
Auto-merging READMETOO
CONFLICT (content): Merge conflict in READMETOO
Automatic merge failed; fix conflicts and then commit the result.

Owner@OWNER-PC ~/desktop/JointProject (master|MERGING)
$ git status
# On branch master
# You have unmerged paths.
#   (fix conflicts and run "git commit")
#
# Unmerged paths:
#   (use "git add <file>..." to mark resolution)
#
#       both modified:      READMETOO
#
no changes added to commit (use "git add" and/or "git commit -a")

Owner@OWNER-PC ~/desktop/JointProject (master|MERGING)
$
```

Move from experiment to master branch.

Merge the experiment into master. Hold on! You've got a conflict. Merger fails until you fix the problem.

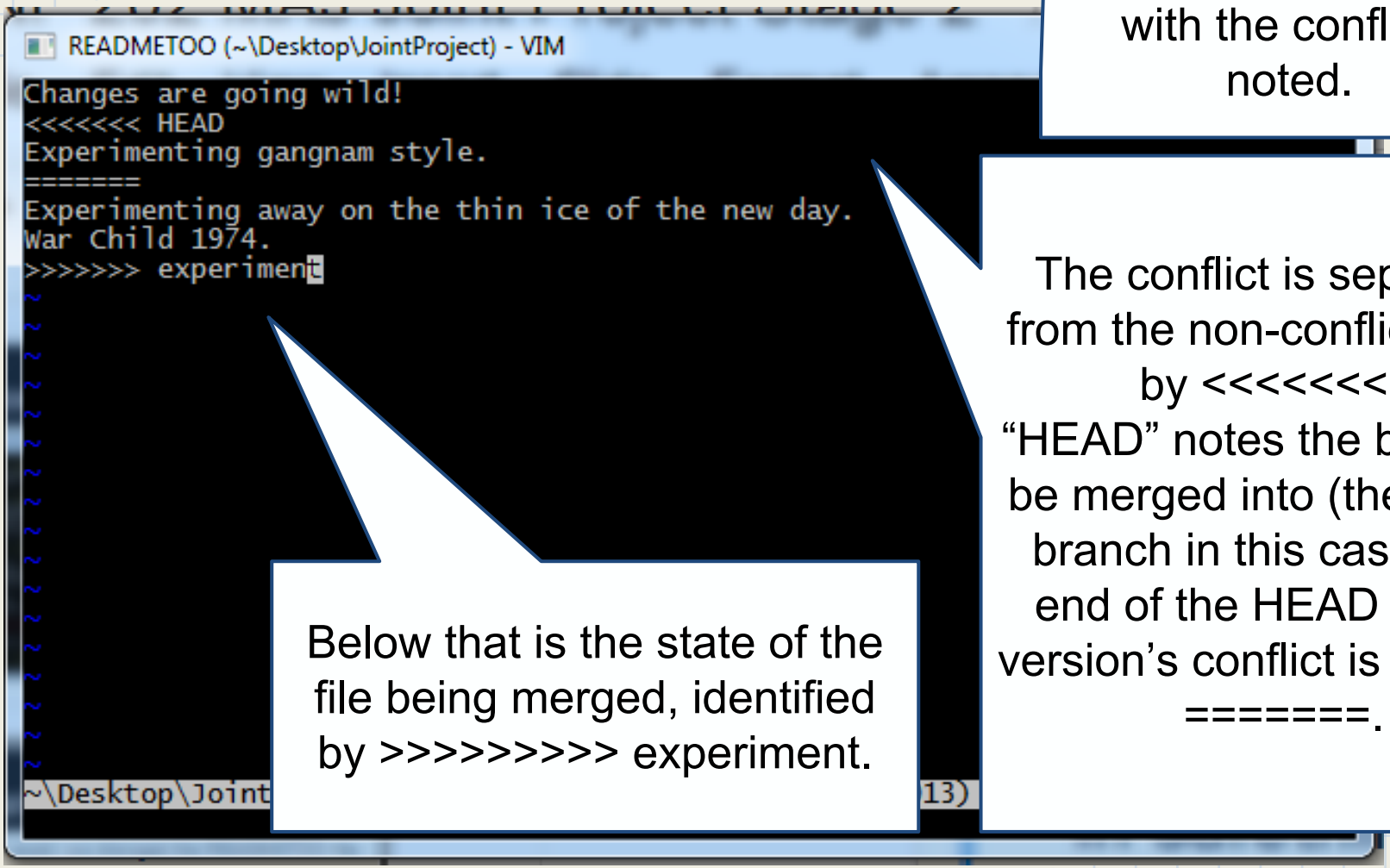
Learn what the problem is using "git status". It's the READMETOO file.

Analyzing the Merge Conflict

- If you recall, we changed the READMETOO file in surething, then merged that change into master (see slides 40-48).
- Then we moved to the experiment branch and changed its version of READMETOO (see 49-51).
- In merging master with experiment, Git found two different versions of READMETOO. Git doesn’t know how to fix that.
- You must open the conflicted file to fix it (vim READMETOO.

Seeing Merge Conflict

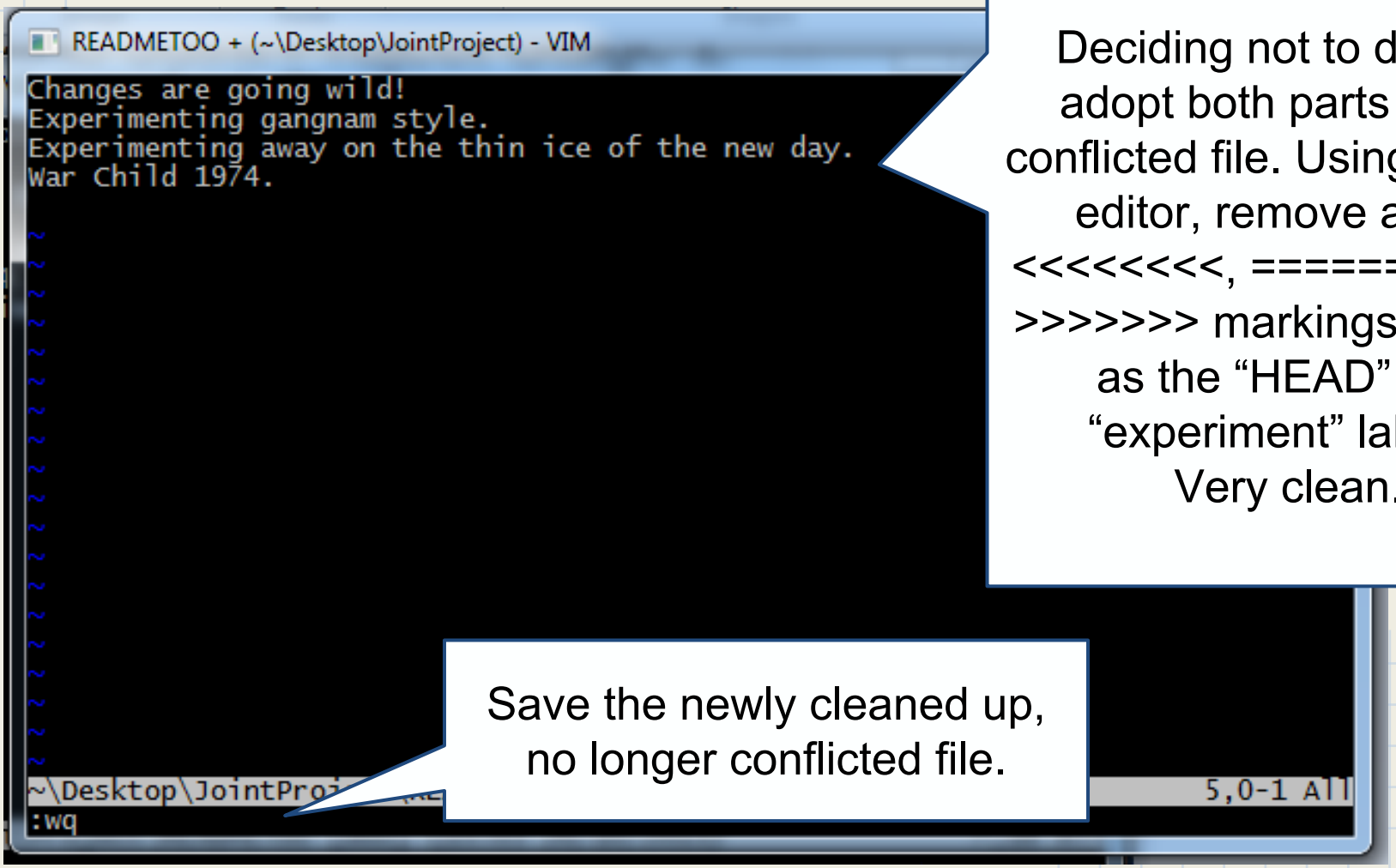
After using “vim READMETOO” to open the file, you see a combined version with the conflicts noted.



The conflict is separated from the non-conflicted text by <<<<<<<<.
“HEAD” notes the branch to be merged into (the master branch in this case). The end of the HEAD branch version’s conflict is noted by =======.

Below that is the state of the file being merged, identified by >>>>>>>>> experiment.

Editing Merge Conflict



Deciding not to decide, adopt both parts of the conflicted file. Using the vim editor, remove all the <<<<<<<, ========, and >>>>>>> markings, as well as the “HEAD” and “experiment” labels. Very clean.

Save the newly cleaned up, no longer conflicted file.

Commit to Resolve Conflict

MINGW32:/c/Users/Owner/desktop/JointProject

```
Owner@OWNER-PC ~/desktop/JointProject (master|MERGING)
$ vim READMETOO

Owner@OWNER-PC ~/desktop/JointProject (master|MERGING)
$ git add READMETOO

Owner@OWNER-PC ~/desktop/JointProject (master|MERGING)
$ git commit_
```

Stage the merge-conflict resolved READMETOO file.

Commit the merge-conflict resolved READMETOO file.

Merge Conflict Resolution Comment

COMMIT_EDITMSG + (~\Desktop\JointProject\.git) - VIM

```
Merge branch 'experiment'
Combined two references (one easy, one obscure), creating snapshot C6.
Conflicts:
        READMETOO
#
# It looks like you may be committing a merge.
# If this is not correct, please remove the file
#       .git/MERGE_HEAD
# and try again.


# Please enter the commit message for your changes. Lines
# with '#' will be ignored, and an empty message aborts t
# On branch master
# All conflicts fixed but you are still merging.
#   (use "git commit" to conclude merge)
#
# Changes to be committed:
#
#       modified:   READMETOO
#
~
~
~
~\Desktop\JointProject\.git\COMMIT_EDITMSG[+] [unix] (15:39 26/10/2013) 2,70 All
:wq
```

Git inserts “Merge branch ‘experiment’” and notes former “Conflicts: READMETOO”. Your comment (red highlighted) explains more.

Save merge conflict resolution commit comment (what a mouthful).

Merge Conflict Resolution Comment

MINGW32:/c/Users/Owner/desktop/JointProject

```
Owner@OWNER-PC ~/desktop/JointProject (master|MERGING)
$ git commit
[master 6670e2a] Merge branch 'experiment' Combined two references (one easy, on
e obscure), creating snapshot C6. Conflicts:    READMETOO

Owner@OWNER-PC ~/desktop/JointProject (master)
$ git log
commit 6670e2a5478482f8416eb0aad45c2a9bb437c723
Merge: 560f052 b9cd67b
Author: M Alexander Jurkat <alex.jurkat@gmail.com>
Date:   Sat Oct 26 15:39:32 2013 -0400

    Merge branch 'experiment'
    Combined two references (one easy, one obscure), creating snapshot C6.
    Conflicts:
        READMETOO

commit b9cd67bc38ebf44ff60a8e128929ff49094c95f5
Author: M Alexander Jurkat <alex.jurkat@gmail.com>
Date:   Sat Oct 26 14:31:58 2013 -0400

    Adding hint to obscure reference, creating snapshot C5.

commit 560f05269f4660e358890a12d6e555b09c267569
Author: M Alexander Jurkat <alex.jurkat@gmail.com>
Date:   Sat Oct 26 12:37:56 2013 -0400

    Can't miss reference in snapshot C4.

commit 958c085f28bc4219e1fb969e2feb4b8d0e0d2ac3
Author: M Alexander Jurkat <alex.jurkat@gmail.com>
Date:   Sat Oct 26 12:04:36 2013 -0400

    Experimenting with README TOO, creating C3

commit 6bc4c06e855c4e62c1cc70110182142bf162e748
Author: M Alexander Jurkat <alex.jurkat@gmail.com>
Date:   Thu Oct 24 23:32:57 2013 -0400

    Changes to two files for snapshot C2.

commit fbd97283f4536cf2e8756e8a02f495e44704c334
Author: M Alexander Jurkat <alex.jurkat@gmail.com>
Date:   Thu Oct 24 22:59:39 2013 -0400

    Changes that create version C1 of project.

commit 3066da9b5c37ba34708b4922618948ae80319218
Author: M Alexander Jurkat <alex.jurkat@gmail.com>
Date:   Sun Sep 22 20:29:54 2013 -0400

    My first commit!
(END)
```

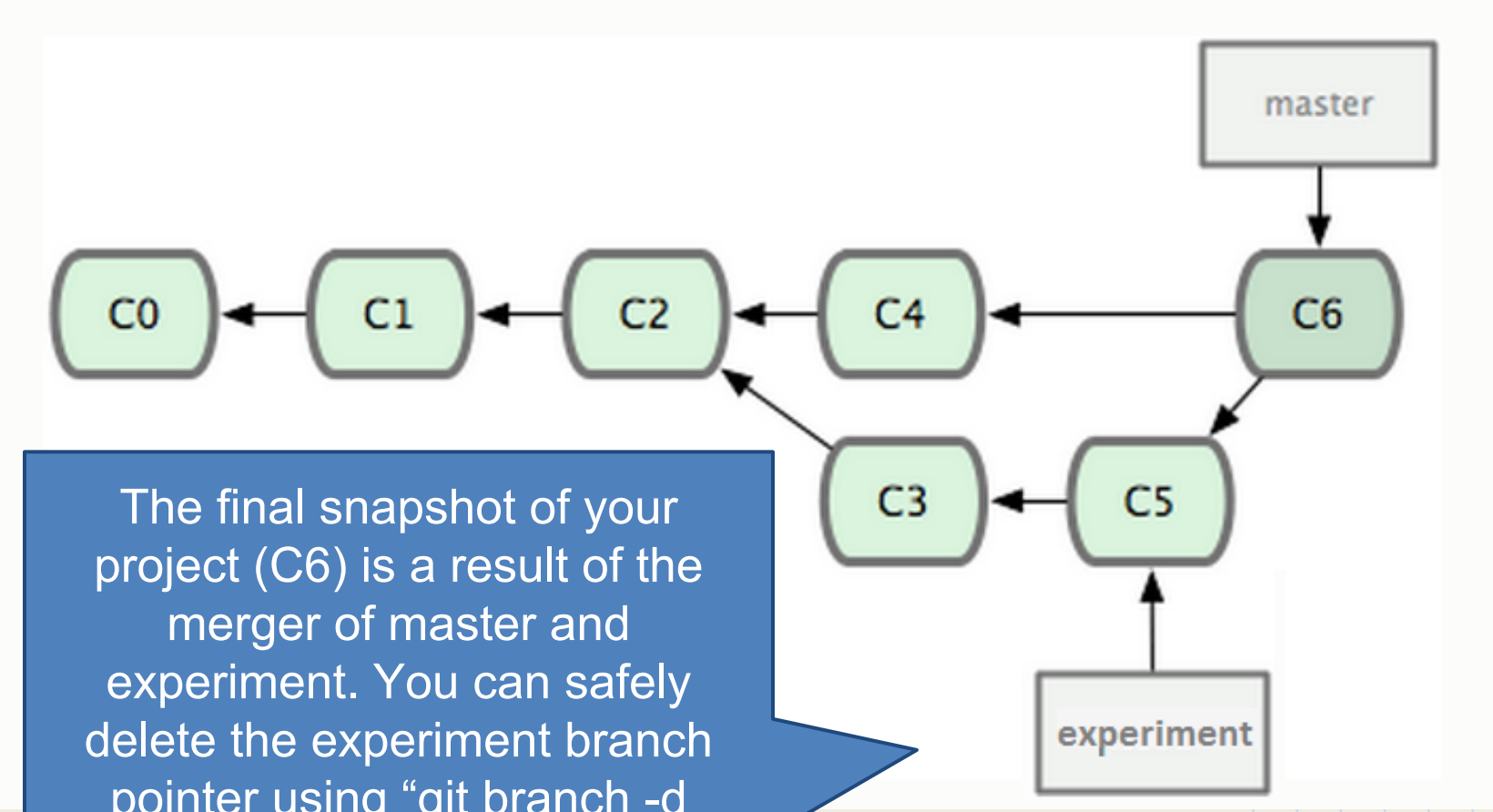
Once you save your commit comment, Git responds that your conflict is resolved.

“Git log” shows the entire history of the project. You may see a “:” at the cursor. Simply press enter to reveal the rest of the log text. Take a screen capture of this log for your Joint Project Stage 2 assignment.

Submit Your Assignment

- Take a screen grab of your final merge conflict resolved commit log message by pressing “Print Scr”.
- Paste your screen grab into MS Paint or its iOS equivalent.
- Save the screen grab as “[yourname] ModuleAssignment1b”.
- Submit the screen grab as an attachment to your Module Assignment 1b submission.

JointProject Snapshot C6



The final snapshot of your project (C6) is a result of the merger of master and experiment. You can safely delete the experiment branch pointer using “git branch -d experiment” if you wish.

That’s a Wrap!

Back to the SMILING!