Technische Universiteit
Eindhoven
University of Technology

Preparing Patches with Git

Roel Jordans

What makes a nice patch

- Describes a **consistent** set of changes to the program
- Has a good explanation of what was added
  - But not necessarily very long
- Easy to verify
  - Apply cleanly
  - Incremental improvement
  - Only changes related things
- Preferably
  - Has accompanying test cases to demonstrate correct functioning (helps verification)

An overview of your changes

- Git keeps a full history of your commits
- “git log” shows an overview
  - Nice list of changes including
    - Who made the commit
    - The commit messages
    - Unique identifiers for each commit

Getting a patch series

- You've learned about git format-patch in the first assignment
- This takes a reference to a commit as argument
  - The HEAD~1
- References can be either
  - HEAD~n, with n the number of changes to go back from the last
    - e.g. "git format-patch HEAD~3" will generate three numbered files
  - A SHA-1 hash identifying the change
    - The really long hexadecimal number which you can find in "git log"

Help, I messed up my patches

- It takes some practice and iteration to split a large change into manageable chunks
- Plan your changes!
- Git allows you to rewrite history

Making last minute additions

- Committed your change and now you notice that it missed something?
  - E.g.
    - A last minute bug-fix in your new code
    - Missing test cases
- “git commit --amend” to the rescue
  - Allows you to add more changes to the previously committed patch

Made more changes to this file?

- You can add a part of the changes in your files
  - Useful if you've started with the next exercise when you notice something was missing in the previous commit
- "git add -p"
  - Asks you for each chunk of changes if it needs to be added to the next commit

Missed things earlier on

- Changing commits before the last one you committed?
- “git rebase -i” allows you to re-order and merge patches
- Requires you to give a commit to start reordering from
  - Needs the same kind of identifier as format-patch
    - HEAD~n or SHA-1 hash

Want to start over?

- If you really messed up you can also have Git forget about your changes
- “git reset”
  - Again needs a reference for which point to reset to
  - e.g. “git reset HEAD~1” forgets about the last commit
- Adding -hard also deletes all changes from files
  - Careful with this, you may lose your code!
- More on git reset

  https://git-scm.com/blog

Technische Universiteit
Eindhoven
University of Technology

Where innovation starts