University of Reading
Department of Computer Science
Pat Parslow, Rhian Taylor, Neyma Siddiqi

Exercise Week 2
CS1PC20 / Autumn 2021
120 Minutes
Discussion date: Week 3

---

1. The exercises are formative – they are there to learn – but also assessed as part of your portfolio of work submitted for course works.
2. Exercises will be discussed on the Week indicated on the header. Additional (non-discussed) training tasks may be provided, and will be shown in red.
3. Make sure to schedule some time each week to keep up with the progress. It is not necessary to complete every task perfectly. If you are struggling, then reach out for help on Teams.

Exercise Introduction

Before attempting the exercises in this document please ensure that you have read and understood the key topics covered in Tutorial 2.

Remember: Tasks count towards your final grade and are also a tool to help you test your own knowledge. You should attempt them as simply learning the theory is not enough, practical application is an invaluable tool in helping you to learn.

Use this exercise to familiarize yourself with git, there should be enough time to get started.

Contents

Task 2: Git Exercises (120 min)

You may want to view the video of me going through the exercise commands before you do it – I was not recording the outcomes, so it is done more quickly than I would expect you to, so the video is only 20 minutes long. You may get some additional hints about what the commands do from the video!

We don't expect for you to be writing long and complex bash scripts straight away. We also appreciate that moving to a CLI (Command Line Interface), while powerful, can be challenging especially when you are accustomed to interacting with primarily graphical interfaces on Operating Systems like Windows.

Below we have included several 'git' commands of varying complexity that also cover more advanced concepts of git. **We want you to experiment with them, try running them and observe the result! Think about what you expect their output to be before you run them.** Perhaps make your own changes to the commands. Use this as an opportunity to learn about got and using it to manage your source code – and other documents.

We would like you to produce a markdown text file providing a short descriptions of your findings, i.e. all you need to provide is a brief description of what the command does (if possible write what you expected) and a short explanation of how it works where appropriate. Marks do not depend on you having the right 'prediction' – so avoid the temptation to run it and then record what happened as what you think will happen!

You should have completed the week 1 exercise first. Normally we would initialise our git repository before creating content, but this exercise assumes you have created content, and shows that we can still set up our repository (normally abbreviated as 'repo') after creating some content.

You do not need to type in the $ at the beginning of each line (in fact, you need to not type it in…)

Where the instruction says <student_id> you should replace that with your student ID

e.g. if your id is xn123456 then
git remote add master https://csgitlab.reading.ac.uk/<student_id>/cs1pc20_portfolio.git

becomes

git remote add master https://csgitlab.reading.ac.uk/xn123456/cs1pc20_portfolio.git

Similarly with <student_email> (use your Reading email) and <student_name> (use your name)

1. $ cd ~
2. $ git init portfolio
3. $ cd portfolio
4. $ ls -al
5. $ git status
6. $ echo hello > .gitignore
7. $ git add -A
8. $ git status
9. $ git config --global user.email "<student_email>"
10. $ git config –global user.name "<student_name>"
11. $ git commit -m "first commit, adding week 1 content"
12. $ git status
13. $ git push

Pause… why did it do that? Read what it says on the screen.

14. $ git remote add origin https://csgitlab.reading.ac.uk/<student_id>/cs1pc20_portfolio.git
15. $ git push –set-upstream origin master

Follow the prompts for username and password – these are your university ID and password

16. $ git status
17. $ echo "# CS1PC20 Portfolio" > readme.md
18. $ git add readme.md
19. $ git commit -m "added readme file"
20. $ git push

Use your web browser to visit https://csgitlab.reading.ac.uk/<student_id>/cs1pc20_portfolio and see what the readme.md file looks like.
If you are like me, you are probably fed up with typing your username and password every time you push, so try this (it stores your username and password in memory for a while the next time you type them in)

21. $ git config –global credential.helper cache
22. $ git branch week2
23. $ git checkout week2
24. $ mkdir week2
25. $ echo "# Week 2 exercise observations" > week2/report.md
26. $ git status
27. $ git add week2
28. $ git commit -m "added week 2 folder and report.md"
21. $ git push
22. $ git checkout master
23. $ ls -al
24. $ git checkout week2
25. $ ls -al
26. $ git checkout master
27. $ git merge week2
28. $ ls -al
29. $ git push
30. $ rm -r week2
31. $ rm -r week1
32. $ ls -al
33. $ git status
34. $ git stash
35. $ git stash drop
36. $ ls -al

The next section makes a copy of your portfolio folder and deletes the original, and then restores it from csgitlab by cloning it. The backup copy made in line 38 is *just in case* you somehow didn't create or can't access the copy on csgitlab!

37. $ cd ~
38. $ cp -r portfolio portfolio_backup
39. $ rm -rf portfolio

40. $ ls -al
41. $ git clone https://csgitlab.reading.ac.uk/<student_id>/cs1pc20_portfolio portfolio
42. $ ls -al

Hints

- Storing your work on CSGitLab lets you move it between computers, and, critically, it is maintained by the University. If it should happen to be unavailable (and this can happen even with e.g. GitHub) then you will have an appropriate automatic extension for coursework. If you store your work on a third party system, or fail to keep a backup at all, and you lose the work or suffer from downtime, there is no extension given.
- Branches let you split work between members of a team.
- Branches also let you try out something experimental without breaking your existing work.
- Git is very powerful and has a lot of ways of returning to previous states. It is generally advised that you do not experiment with these on a project unless you know you have a backup you can revert to, as it can be a bit 'tricky' (even people used to using git will often have to refer to others to remember exactly how to recover from 'situation X' …)
- Do watch the various videos provided which give examples of using git and moving documents etc.
- Use the commands
  $ git --help
  $ git branch –help
  $ git checkout –help
  (and so on) to find out what a specific git command does.

Portfolio (directory: portfolio/week2/introductory-git.md)

portfolio/week2/introductory-git.md
: An enumerated list that provides for each command a prose sentence what the commands did. Where your expectation differs from observed behaviour, include a sentence describing what surprised you. See if you can explain what the commands did (briefly!)

Further Reading

- An introduction to Got https://www.freecodecamp.org/news/what-is-git-and-how-to-use-it-c341b049ae61/
- Git tutorial: https://git-scm.com/docs/gittutorial
- Interactive git branching tutorial: https://learngitbranching.js.org/