Summary: All students who have been assigned to a group for their ENCE260 project AND logged into the eng-git server should have received an email stating that they can now log into git and use this as part of their code development. This note provides information on how you and your project partner can start using your newly created git repository. It is by no means meant to be a comprehensive guide. This note is designed to get you started so you can at least push your source files to your **eng-git** repository and keep this updated. There will also be one lecture as an overview of git.

1. Introduction

git is a distributed revision control system. This means every working directory is a fully-fledged repository with complete history and full revision tracking capabilities, and is not dependent on network access or a central server. If you haven't used a distributed revision system before the process can be a little intimidating. This guide will hopefully take you through most of the pitfalls and at least get you started.

Here are a few reasons why you should use git for your assignment:

1. Both you and you project partner can work collaboratively on code development.
2. Updates can periodically be pushed to the **eng-git** server. Local commits are saved to your local drive or server. Past revisions are never lost and can be easily retrieved.
3. Two levels of file management and revision control are available - local and server based.
4. Use of a repository such as git prepares students for revision control used in industry.
5. Last, but most importantly, we will only be able to access and mark your code if it is in your group's git repository! Both group members must make multiple commits for full marks.

2. Command line tutorial

So, you have received an email from gitlab saying something like: "You have been granted Developer access to group ence260-2025/GroupXXX. If you are in the lab, your PC has git installed. If you are using your own computer, you may need to install git.

The following tutorial assumes you have: i) an internet connection, ii) you have a command line version of git installed and available, and iii) you are logged into your terminal program and in your `../ence260-ucfk4/assignment` subdirectory. This last requirement is particularly important.

1. Type in: "git clone https://eng-git.canterbury.ac.nz/ence260-2025/GroupXXX.git", where XXX is your group number. If all is well, you should see a message in your terminal window saying something like "Cloning into 'GroupXXX.git' …" and a warning message that you have cloned an empty repository. This warning is fine. We have just created your GroupXXX repository so there will be nothing in it. Pushing your source files into this will be up to you!
2. Check that you have a newly created "GroupXXX" directory by typing "ls" at the command prompt. You should see: "README template GroupXXX".
3. Now copy some files into your newly created GroupXXX directory. To get you going, you may want to copy the Makefile and game.c source files from the /template directory. These are just very simple example files in an example directory. The source, game.c, should compile in your "/template" directory, and if you move these files they should also compile in your "/GroupXXX" directory. Make sure they do compile by typing in "make". They won't do much but, again, they're not meant to.
4. We are now going to add these files to your git repository. You only need to do this once. To do this simply type: "git add game.c". You may not get any feedback from this. Then type in: "git add Makefile".
5. Use "git status" to ensure your source files have been added and updated.
6. The next step is to commit your source files to your repository. Do this by typing in the following:

   git commit game.c -m "make a comment on the major changes you have made"

Note that the double quotes are required!

7. Also do a commit for your Makefile. Remember, your Makefile is also a source file. Note that you should receive feedback here, such as “one file changed ...”
8. The last thing for you to do is to push your last commits on to your GroupXXX repository on our eng-git server. To do this type the following:

   git push origin main

   Once, you have typed and entered this in the command window you should see some dialogue from git stating something like:

```
rmc84@ELECST024 MINGW64 /c/Users/rmc84/Dropbox (UC Enterprise)/TEACHIN
$ git push origin main
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 8 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (4/4), 821 bytes | 821.00 KiB/s, done.
Total 4 (delta 0), reused 0 (delta 0), pack-reused 0
To https://eng-git.canterbury.ac.nz/ence260-2023/group_000.git
 * [new branch]      main -> main
```

9. That’s it for now. At this point you should check your git repository and verify that your files have indeed been pushed onto our eng-git server. You may even like to download your files, which will be nicely wrapped in a git wrapper! More importantly, your project partner will be able to clone these files onto their PC or Laptop. This is where you start to work collaboratively together on code development. Both of you can push your work to update your shared repository, however, and most importantly, execute a git pull first in order to ensure your code is updated and synchronised with your code repository before you submit your changes. If no changes have been made by your project partner, you should receive the message, “Already up-to-date”.

3. Traps and pitfalls

1. Use the recommended directory for your game program development, such as:

   `../ence260-ucfk4/assignment/GroupXXX`

   where XXX stands for your group number. If you don’t then the Makefile paths will not be valid.
2. Commit regularly, i.e., every hour or so. Push every session. Again, you have a two tier revision tracking system, however, you will only update your partner’s code when you pull their code from the eng-git server first.
3. Do not include the API functions (eg from /drivers or /utils) in your repository or any object (.o) or executable (.out or .hex) files. You should only include the Makefile and source files that you have written.
4. Do not create subdirs from your /GroupXXX dir. All code should be in /GroupXXX.
5. If you create branches in your repository, only the default branch will be cloned and marked

4. Documentation

Documentation for the UCFK4 can be found on Learn (under ENCE260, Embedded Systems, UCFK: Fun Kit) at:

https://learn.canterbury.ac.nz/mod/page/view.php?id=4217084