

ECSE 321 Introduction to Software
Engineering
Hands-on Tutorials

McGill University

Table of Contents

1. Preliminaries	1
1.1. Getting Started.....	2
1.2. Project Management Tools for Agile Development	3
1.2.1. GitHub Projects	3
1.3. Command Line Basics	6
1.3.1. Windows prerequisites	6
1.3.2. Basic file system operations.....	6
1.3.3. Finding files	8
1.3.4. Batch file operations	8
1.3.5. Some additional useful commands.....	8

- [HTML version](#)
- [PDF version](#)

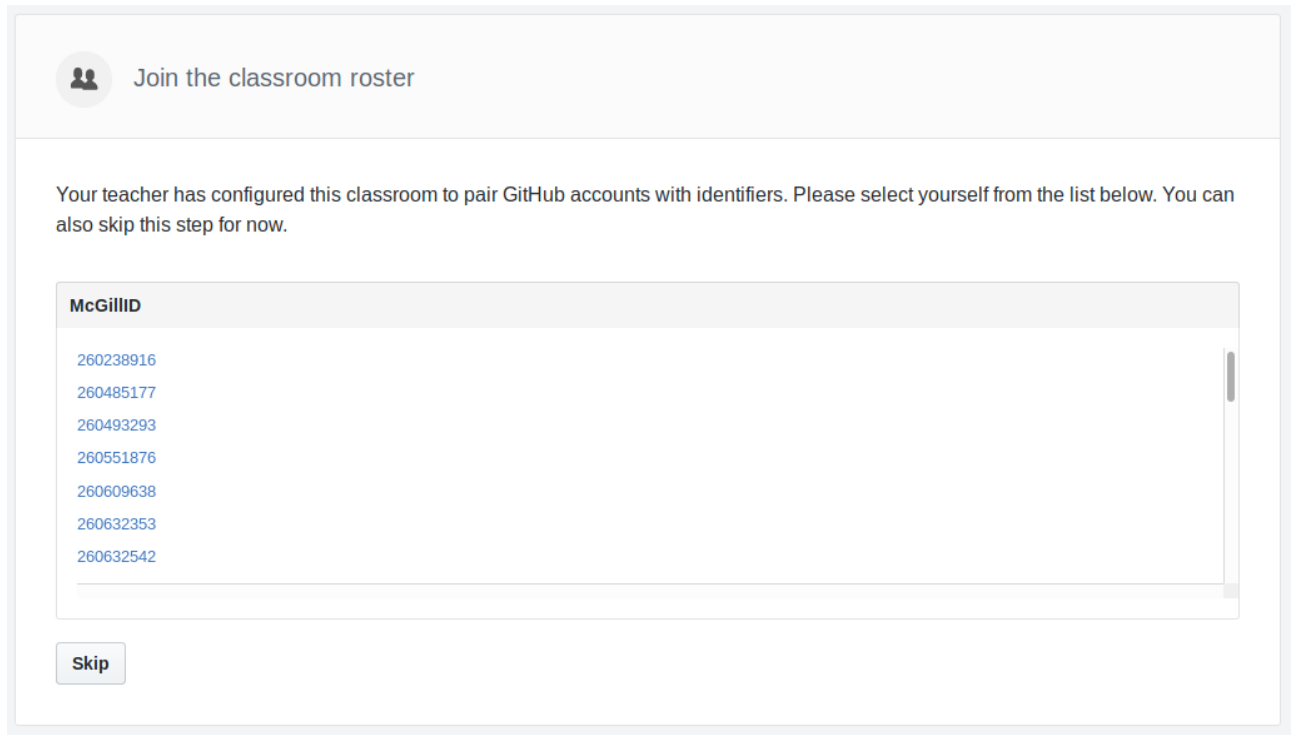
Sections of the tutorial will continuously be published at this web page.

1. Preliminaries

1.1. Getting Started

Steps for signing up for GitHub classroom:

1. Log in/Register on GitHub.
2. Open link <https://classroom.github.com/g/o9gWNZis>
3. Select your McGill ID from the list



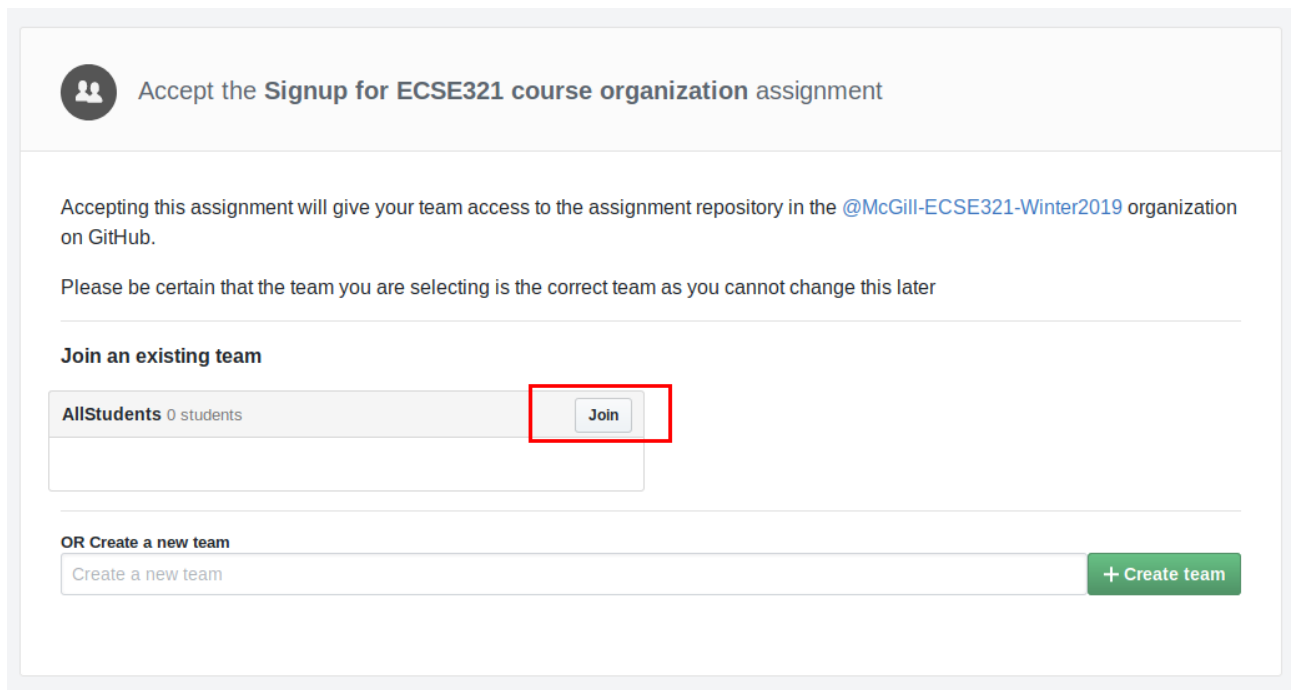
Join the classroom roster

Your teacher has configured this classroom to pair GitHub accounts with identifiers. Please select yourself from the list below. You can also skip this step for now.

McGillID
260238916
260485177
260493293
260551876
260609638
260632353
260632542

Skip

4. Join team *All students*



Accept the Signup for ECSE321 course organization assignment

Accepting this assignment will give your team access to the assignment repository in the [@McGill-ECSE321-Winter2019](#) organization on GitHub.

Please be certain that the team you are selecting is the correct team as you cannot change this later

Join an existing team

Team Name	Students	Join
AllStudents	0 students	Join

OR Create a new team

Create a new team **+ Create team**

1.2. Project Management Tools for Agile Development


1.2.1. GitHub Projects

First, we create a new repository under everyone's own account to demonstrate the basic features of "GitHub Projects".

1. Visit <https://github.com/> then click on *New repository* (green button on the right).
2. Set your user as the owner of the repository.
3. Give a name for the repository (e.g., ecse321-tutorial-1), leave it *public*, then check *Initialize this repository with a README*. Click on *Create repository* afterwards. At this point the remote repository is ready to use.


Create a new repository


A repository contains all the files for your project, including the revision history.

Owner	Repository name
 ecse321testuser ▼	/ ecse321-tutorial-1 ✓

Great repository names are short and memorable. Need inspiration? How about **furry-octo-journey**.


Description (optional)

☒  **Public**
Anyone can see this repository. You choose who can commit.

☐  **Private**
You choose who can see and commit to this repository.

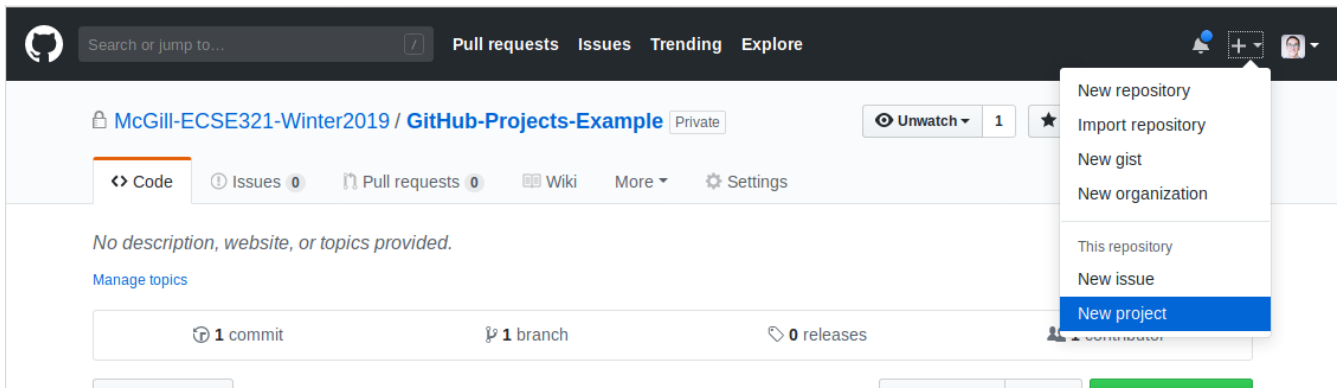
☒ **Initialize this repository with a README**
This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

Add .gitignore: **None** ▼

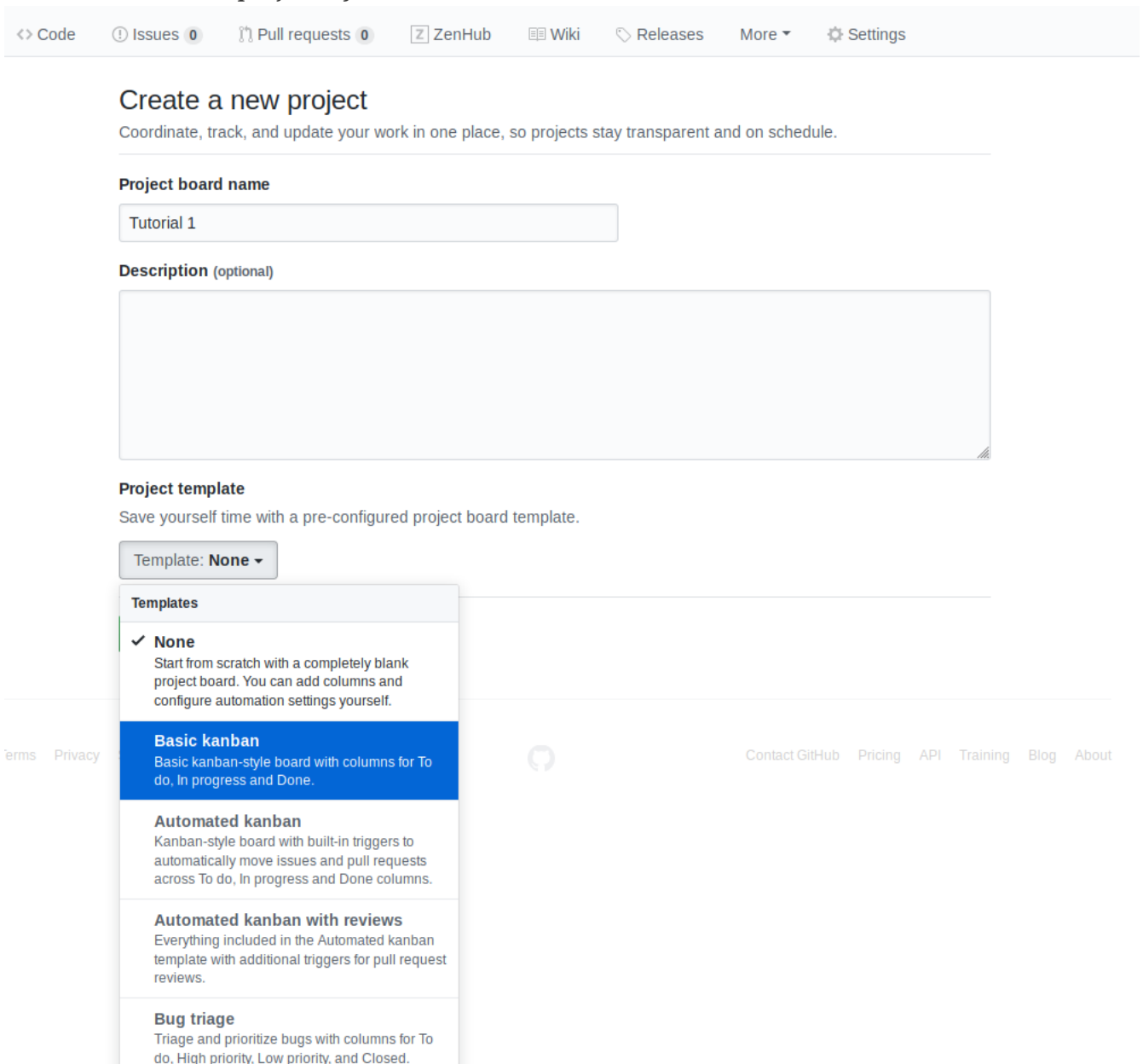
Add a license: **None** ▼ 

Create repository

Once the repository is ready, associate a new GitHub Project and see how their features work. Create a project:



Select Basic Kanban project style:



Tasks to complete:

1. Create a few issues to outline the tasks for the first deliverable. Assign them appropriate labels and add yourself as the assignee!

[Code](#)
[Issues 5](#)
[Pull requests 0](#)
[ZenHub](#)
[Projects 1](#)
[Wiki](#)
[Releases](#)
[More](#)
[Settings](#)

Filters
[Labels](#)
[Milestones](#)
[New issue](#)

[Clear current search query, filters, and sorts](#)

<input type="checkbox"/>	5 Open	0 Closed	Open All	Author	Labels	Projects	Milestones	Assignee	Sort
<input type="checkbox"/>	Create UML Class diagram in UML Lab								1
	#1 opened 2 days ago by imbur updated 2 days ago								
<input type="checkbox"/>	Add UML Diagram								
	#2 opened 2 days ago by imbur updated 2 days ago								
	documentation								
<input type="checkbox"/>	Create database layer								
	#5 opened 2 days ago by imbur updated 2 days ago								
	epic								
<input type="checkbox"/>	Write project deliverable 1								
	#4 opened 2 days ago by imbur updated 2 days ago								
	epic								
<input type="checkbox"/>	Report individual and teamwork								
	#3 opened 2 days ago by imbur updated 2 days ago								
	documentation								

2. Create a milestone for the issues.

[McGill-ECSE321-Winter2019 / GitHub-Projects-Example](#)
[Private](#)
[Unwatch 1](#)
[Star 0](#)
[Fork 0](#)

[Code](#)
[Issues 5](#)
[Pull requests 0](#)
[ZenHub](#)
[Projects 1](#)
[Wiki](#)
[Releases](#)
[More](#)
[Settings](#)

[Labels](#)
[Milestones](#)
[New milestone](#)

0 Open 0 Closed
 [Sort](#)

3. Create cards from the issues on the project board.

4. See how GitHub track the project progress as you move the cards from the different columns.

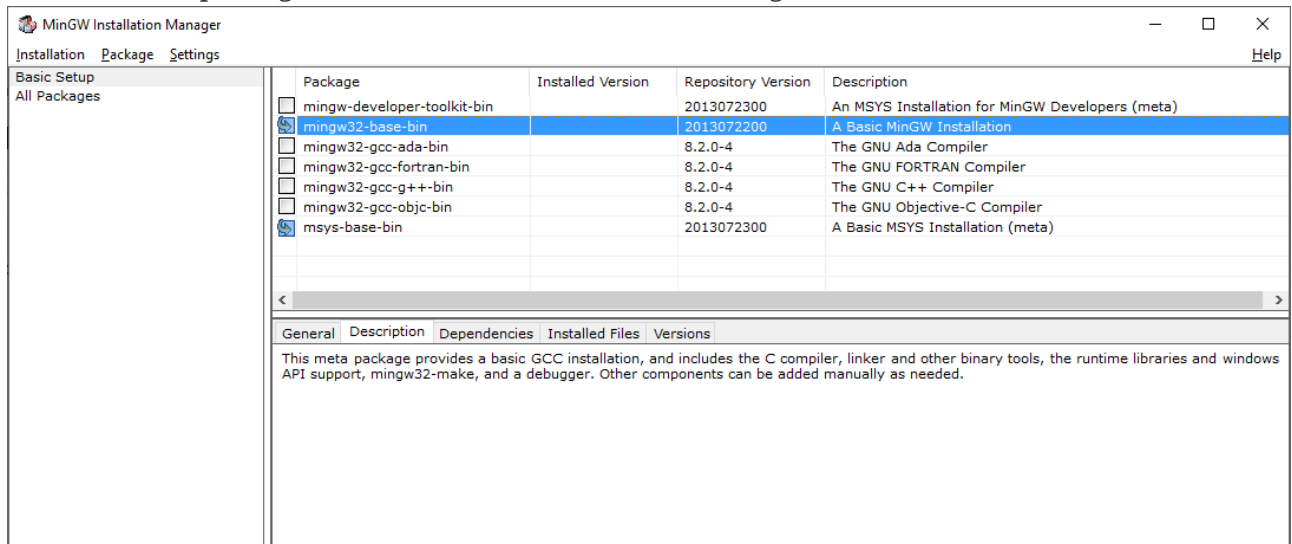
1.3. Command Line Basics

This section shows a few handy terminal commands.

1.3.1. Windows prerequisites

This step can be skipped if you are using MacOS or Linux. However, if you are using Windows, you need to have a terminal that supports the execution of basic Linux commands. Such programs are Git Bash or MinGW, for example. You can find below a few helper steps to get MinGW running on your system.

1. Get the [MinGW installer from here](#)
2. Install it to wherever you like, the default installation folder is `C:|MinGW`
3. Once the setup finishes, open the MinGW Installation Manager
4. Select the two packages for installation as shown in the figure below



5. Click on *Installation/Apply Changes*. This will take a few moments to fetch and install the required packages.
6. You can open a terminal window by running the executable `C:|MinGW|msys|1.0|bin|bash.exe`

1.3.2. Basic file system operations

1. Open a terminal, and try the following commands:

- `pwd`: prints the present working directory

Example:

```
$ pwd
/home/ecse321
```

- `ls`: lists the content of a given folder

Example:


```
$ ls /home
ecse321 guest-user admin
```

- **cd**: navigates the file system

Example:

```
$ cd ..
$ pwd
/home
$ cd ecse321
$ pwd
/home/ecse321
```

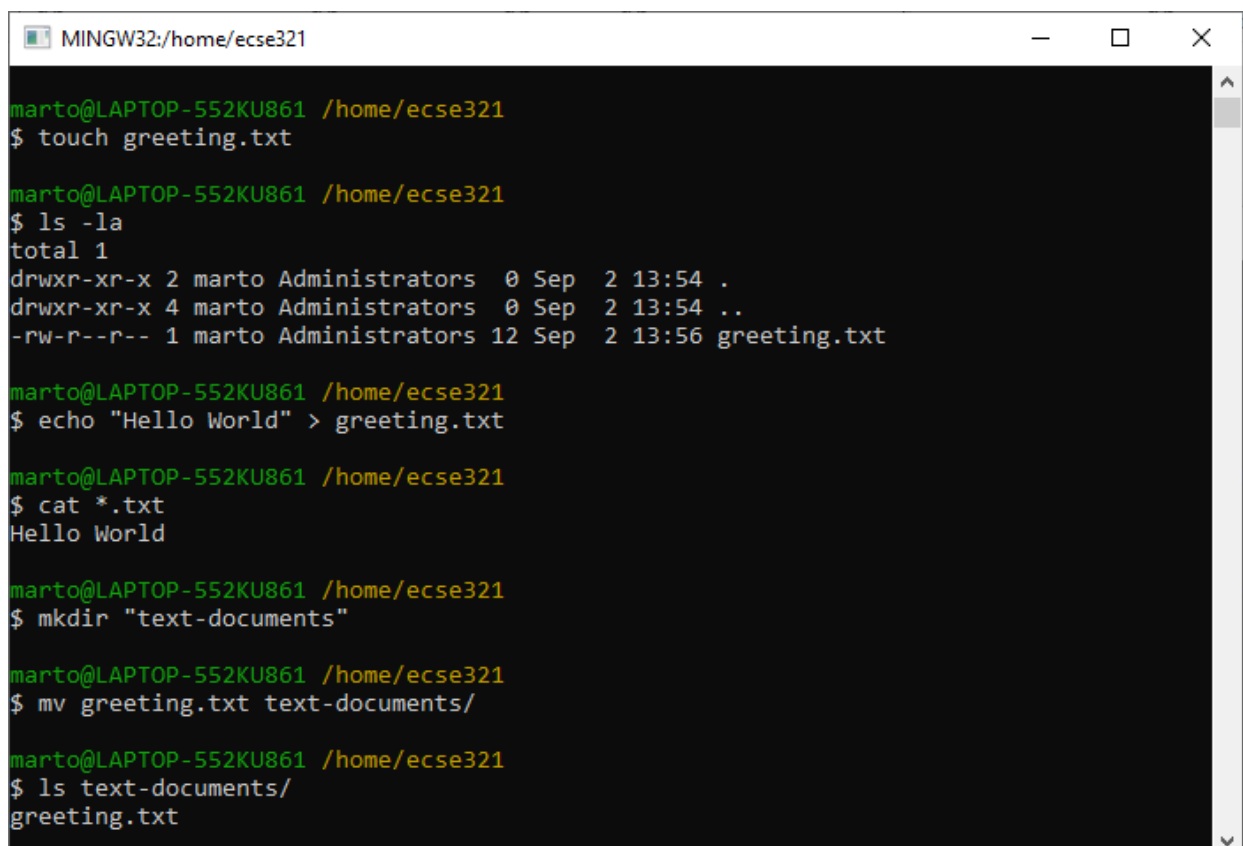
NOTE

The following steps will include images that illustrate the commands and their output to prevent easy copy-paste. Sorry! :)

2. Creating files and reading/writing their contents

- **touch**: creates a file
- **mkdir**: creates a directory
- **mv**: moves a file (or directory) from its current location to a target location
- **echo**: prints a string
- **cat**: prints the contents of a file

Example:



```
MINGW32:/home/ecse321

marto@LAPTOP-552KU861 /home/ecse321
$ touch greeting.txt

marto@LAPTOP-552KU861 /home/ecse321
$ ls -la
total 1
drwxr-xr-x 2 marto Administrators  0 Sep  2 13:54 .
drwxr-xr-x 4 marto Administrators  0 Sep  2 13:54 ..
-rw-r--r-- 1 marto Administrators 12 Sep  2 13:56 greeting.txt

marto@LAPTOP-552KU861 /home/ecse321
$ echo "Hello World" > greeting.txt

marto@LAPTOP-552KU861 /home/ecse321
$ cat *.txt
Hello World

marto@LAPTOP-552KU861 /home/ecse321
$ mkdir "text-documents"

marto@LAPTOP-552KU861 /home/ecse321
$ mv greeting.txt text-documents/

marto@LAPTOP-552KU861 /home/ecse321
$ ls text-documents/
greeting.txt
```

1.3.3. Finding files

The versatile `find` command allows us to find files based on given criteria. Take look at its manual page with `man find`!

Example:

```
MINGW32:/home/ecse321
marto@LAPTOP-552KU861 /home/ecse321
$ ls -la
total 0
drwxr-xr-x 3 marto Administrators 0 Sep  2 23:05 .
drwxr-xr-x 4 marto Administrators 0 Sep  2 13:54 ..
drwxr-xr-x 2 marto Administrators 0 Sep  2 23:05 text-documents

marto@LAPTOP-552KU861 /home/ecse321
$ find ./ -iname *.txt
./text-documents/greeting.txt
```

1.3.4. Batch file operations

- `sed`: stream editor; changes a given string to a replacement

Combining `find` with an additional command (e.g., `sed`) can greatly speed up your repetitive tasks.

Example:

```
MINGW32:/home/ecse321
marto@LAPTOP-552KU861 /home/ecse321
$ ls -la text-documents/
total 2
drwxr-xr-x 2 marto Administrators  0 Sep  2 23:26 .
drwxr-xr-x 3 marto Administrators  0 Sep  2 23:05 ..
-r--r--r-- 1 marto Administrators 14 Sep  2 23:26 greeting.txt
-rw-r--r-- 1 marto Administrators 12 Sep  2 23:21 helloworld.txt

marto@LAPTOP-552KU861 /home/ecse321
$ touch temp

marto@LAPTOP-552KU861 /home/ecse321
$ sed "s/World/ECSE321/g" text-documents/greeting.txt temp
Hello ECSE321

marto@LAPTOP-552KU861 /home/ecse321
$ cat temp
Hello ECSE321

marto@LAPTOP-552KU861 /home/ecse321
$ sed "s/World/ECSE321/g" text-documents/greeting.txt > temp

marto@LAPTOP-552KU861 /home/ecse321
$ cat temp
Hello ECSE321

marto@LAPTOP-552KU861 /home/ecse321
$ mv temp text-documents/greeting.txt

marto@LAPTOP-552KU861 /home/ecse321
$ find ./ -iname *.txt -exec sed "s/Hello/Hi/g" {} \;
Hi ECSE321
Hi World
```

NOTE The file *helloworld.txt* in the example is initially a copy of *greeting.txt*.

1.3.5. Some additional useful commands

- `rm`: removes a file
- `cp -r`: copies a directory recursively with its contents
- `rmdir`: remove an empty directory

- `rm -rf`: force to recursively delete a directory (or file) and all its contents
- `nano`: an easy-to-use text editor (not available by default in MinGW)
- `grep`: finds matches for a string in a given stream of characters
- `ag`: takes a string as argument and searches through the contents of files recursively to find matches of the given string (this tool is included in the *silversearcher-ag* package)