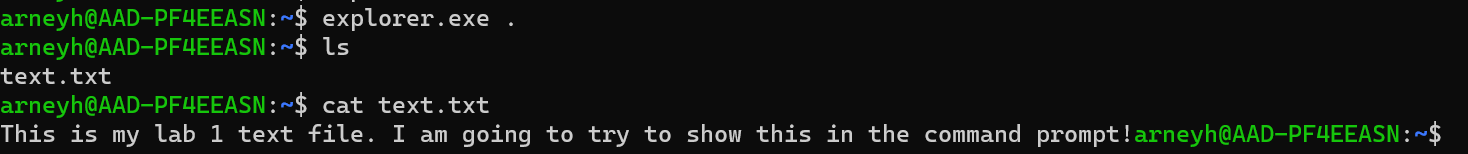
# Lab Report 1 – Introduction to MySQL

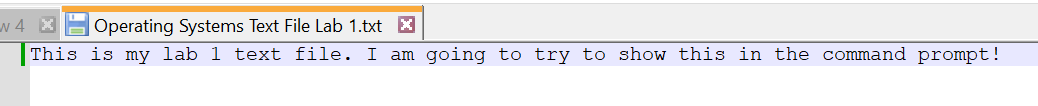
## Operating Systems - CSC 3210 003

### Hudson Arney

### 23 January 2024

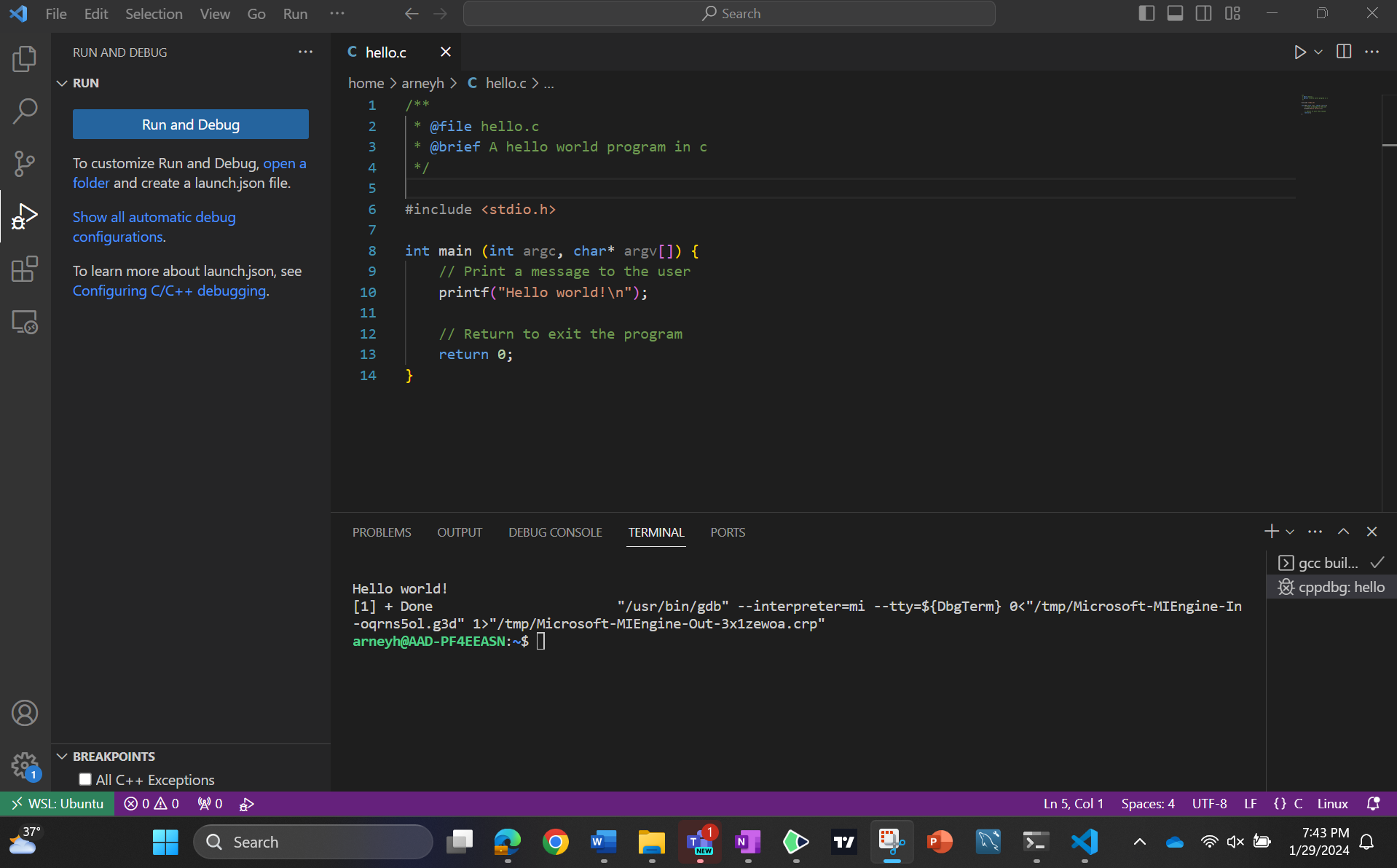
Editing a TextFile





Exploring Linux

* ls – lists out the options you are looking for in each command. This could be a list of possible instructions you can use, or the files within your current directory, etc.
* find – allows you to find a specific file through a search.
* cd – allows you to navigate to different directories in file explorer.
* mkdir – lets you create a new directory that starts in your current directory.
* rmdir – lets you remove the directory (you must specify the name).
* cp – lets you copy files from one directory to another, but you must specify the destination and source.
* cat – will display the contents of the specified file in the terminal.
* zip – will configure specified files into a zip format (compression of files).
* tar – I wasn’t sure what a tarball was before reading but it seems to be a compression similar to zipping. It lets you compress a group of directories or files.
* touch – creates an empty file that you have to name.



* What files do you see? Include a 'screenshot' of the listing output.

I’m not in a specific directory, I’m just in arneyh which is the standard directory the terminal goes for. I created the c file in vscode and ran it through the command line. You can see the c file 5th from the bottom.

A screenshot of a computer

Description automatically generated

* How big is each file? How did you determine the size(s)?

Each file has its own size (which I’m assuming are in bytes) but there are a lot of 4096 which I believe are for empty folders. The c file has one of the smallest storages here. I am getting them from the 5th column in the screenshot.

* What are the permissions on each file? How did you determine these permissions? Do the permissions make sense? Why or why not?

The first column in the screenshot show the permissions of the file. “Rwx” stands for read, write, execute. In the example above, a.out and c file have read, write, and execute permissions for the owner, group, and others (-rwx).

* Research the compiler gcc, what file contains the executable generated from compiling hello.c?

The executable generated is the a.out file that can be seen in the screenshot two rows above the c file itself. It has more storage than the actual c file.

Working with Linux Files & Permissions

* **Regular File –** The large majority of the files found on UNIX and Linux systems. Typically, these files contain ASCII (human-readable) text, executable program binaries, program data, etc.
  + **Use/Example:** Most files that hold data which could be in the form of an image, document, music, etc.
* **Directory –** Is a binary file used to track and locate other files and directories. The binary format is used so that directories containing large numbers of filenames can be search quickly.
  + **Use:** Directories are used to organize and store files. They contain references (entries) to other files and directories.
* **Symbolic Link –** Contain a pointer, or pathname, to the original file. If the original file is deleted, its data can no longer be accessed using the symbolic link, and the link is then considered to be a stale link.
  + **Use:** Symbolic links are shortcuts or references to another file or directory. They provide a convenient way to access a file or directory from a different location in the file system.
* **First In/First Out - a.k.a Named Pipe –** allow two or more system processes to communicate with each other using a file that acts as a pipe between them. This type of communication is known as interprocess communication (ICP).
  + **Use:** For communication between two processes by reading and writing to a common file.
* **Block Special –** Used for device I/O, data is transferred in large fixed-size blocks. This type of access is called block device access.
  + **Use:** Block special files represent block devices, typically used for random-access devices like hard drives and SSDs.
* **Character Special** – Also used for device I/O, data is transferred one character at a time. This type of access is called raw device access.
  + **Use:** Often used for devices that deal with streaming data character by character such as a keyboard or serial port.
* **Socket –** Sockets are also tools used for interprocess communication; however, sockets will facilitate communication between processes running on different systems, or over the network.
  + **Use:** Sockets enable communication between processes on different computers or within the same computer.

Information provided from here:

* [File Types in Unix: Ordinary or Regular Files, Directories, Device (Special) Files, Links, Named Pipes, and Sockets (livefirelabs.com)](https://www.livefirelabs.com/unix_tip_trick_shell_script/unix_operating_system_fundamentals/file-types-in-unix.htm)
* [Unix File System - GeeksforGeeks](https://www.geeksforgeeks.org/unix-file-system/)

Working with File/Directory Permissions

1. Write the ***exact*** command necessary to perform the long listing. Capture a screenshot of the long listing output.

A screen shot of a computer screen

Description automatically generated

**ls -l**

1. Research file permissions and how to change them.
   1. What is the difference between user, group, and other permissions? What priority is used to apply the permissions when accessing a file? The user is like the owner of the file. The group is a set of users that belong to the same group. And other is any other user outside the group. User takes precedence over group which takes precedence over other.

[Precedence of user and group owner in file permissions - Unix & Linux Stack Exchange](https://unix.stackexchange.com/questions/134332/precedence-of-user-and-group-owner-in-file-permissions#:~:text=User%20takes%20precedence%20over%20group%20which%20takes%20precedence,is%20used%20to%20determine%20whether%20access%20is%20granted.)

* 1. What file permissions need to be set or removed to allow all others to read a file in a directory? Remembering the Read, Write, Execute permissions from earlier, you would need to set all Others permission of reading in the directory to on. You could do it with a command like this: chmod o+r filename
  2. Change the file permissions on the compiled executable created from hello.c (most likely a.out) to remove the execute permissions from the user, group, and others. You would have to switch the executable permissions for the a.out file to off. You can do this with the command: chmod a-x a.out

The a in the command stands for all (User, Group, and Others) -x is for removing executable permissions.

* 1. Write the ***exact*** command necessary to change these permissions. Command is above

[How to Use the chmod Command on Linux (howtogeek.com)](https://www.howtogeek.com/437958/how-to-use-the-chmod-command-on-linux/)

* 1. Run the executable now that the file permissions are changed. Did it execute as it did before? Why or why not? Research file permissions to find out. Make sure to cite your sources

No it did not execute as before and says “Permission Denied.” Because we removed permissions, we can no longer execute the executable file a.out. I check the permissions using ls -l a.out and the first column is no longer rwxrwxrwx.

1. Create a directory - call it whatever you like. What are the default permissions? drwxrwxrwx – the d stands for directory, the rest just has the read, write, execute permissions for users, groups, and others.
   1. Research directory permissions - What does it mean to have read, write, and execute permissions on a directory? The read permission lets you view or read the directory. The write permission lets you write or modify the directory. The execute permission lets you search a directory.

[Basic Linux Directory Permissions and How to Check them (rackspace.com)](https://docs.rackspace.com/docs/basic-linux-directory-permissions-and-how-to-check-them)

* 1. What permissions need to be set or removed to prevent a user from creating files in a directory?

You would need to remove the write permissions for that user on that directory. Again you can use the chmod o-w directory\_name command.

* 1. What are 'pass-through' permissions?

They allow a user to pass through a directory to other subdirectories.

Working with Symbolic Links

1. Research symbolic links. What is the ***exact*** command needed to create a symbolic link to your hello executable? Call it hello\_link. ln -s hello hello\_link. This worked when I was in the same directory as hello.c, otherwise it will make a new blank file in the given directory called “hello.”
2. Perform a 'long listing' of the directory containing your link. What are the file permissions on the link. Include a screenshot of this listing in your submission.

A screenshot of a computer

Description automatically generated

l which stands for link, and then all the permissions for every user.

1. Research link permissions - what do permissions mean to a symbolic link? Does this make sense to you? Why or why not?

Permissions on a symbolic link don’t mean much, what really matters are the permissions on the file that it points to (in this case hello.c).

This makes sense to me because the link is just an access point to get to the file it points at, you don’t really need to read or write to it.

<https://www.bing.com/search?pglt=675&q=3.+Research+link+permissions+-+what+do+permissions+mean+to+a+symbolic+link%3F&cvid=e7d8ee6a6d1b4e8a9fd203916a4c1a5c&gs_lcrp=EgZjaHJvbWUyBggAEEUYOTIHCAEQRRj8VdIBBzIwMmowajGoAgCwAgA&FORM=ANNAB1&PC=U531>

1. What file types can be the target of a symbolic link? Can symbolic links be the target of other symbolic links? How does Linux handle the case where a link target has been removed but the symbolic link still exists?

Any file or directory can be pointed to by a symbolic link.

<https://stackoverflow.com/questions/4695241/does-creating-a-symbolic-link-to-another-symbolic-link-have-any-side-effects>

You can create a symbolic link to another symbolic link, it is called a chained symbolic link but this doesn’t have many practical uses and can be confusing.

If you delete a file for which a symbolic link still exists, the rm command will succeed but the symbolic link would remain. Any attempt to reference it will return a ‘file not found’ error.

<https://superuser.com/questions/1170056/unix-why-soft-link-file-still-pointing-to-original-file-when-original-file-is-d>

Suggestions:

I personally thought this lab was pretty long, however, I really liked getting to try a lot of this on my own. Most of my time got spent doing less important parts of the lab such as setting up the compiler for C files in VSCode and forgetting a period in the execute statement for the textfile part. Overall, I do feel a lot more comfortable in Linux as a result of this lab but the extent I feel took longer than need be.