**Student Name: Demetrius Johnson**

**Course: CIS-450-002**

**Professor: Dr. Jinhua Guo**

**Date: March 09, 2022**

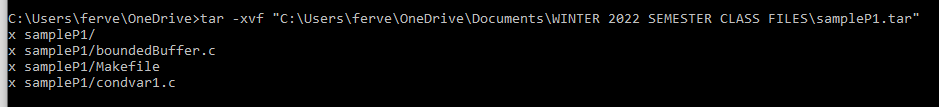
**Due Date: March 14, 2022**

**Project 3 Report**

# Question 1: Producer and Consumer Problem

# Additional Notes

* If path name is not working when using a command, use quotations around the *entire*  file path name. Below is an example in Windows OS terminal:



* Make files: a file that contains a set of commands, that will be issued and expected to do work on (such as compilation, deletion, etc.) a set of files in the directory where the make file resides. It is essentially a *function*, or more precisely, a *parameter* to a built in Linux OS system function.
  + Note: make files are optional and customizable to make it easier for a programmer to delete or compile and link multiple programs/files using a single command. It is essentially a parameter to a OS function that allows you to issue a summary of OS commands all in one command. See example below:
  + Text

    Description automatically generated
  + Above, the last line is the name of the executable that the professor runs.
  + $make clean //finds the make file in the current directory and passes it to the Linux OS system function; clean is another parameter, a reference to a function inside of the make file that will also be passed to the OS system function.
    - The line below (rm bound…) shows the commands issued by the “make clean” command. It removes all files with the given names that contain .o file extension.
  + $make //finds the make file in the current directory and passes it to the Linux OS system function; notice the four lines/commands issued below it represent the summary of functions/commands/parameters called by the make file/parameter. In this case, the make file compiles and links 2 separate programs (boundedBuffer and condvar1). -lpthread and -lrt are native/default c-libraries to which the .o files are linked to before generating the executable. You can also link other object files simply by using *filename.o*.
    - **g++ -c file\_name** is used to only compile and assemble the **file\_name** and **not** link the object code (.o file) to produce executable file. It will generate a **file\_name.o** object code file in present working directory.
    - **g++ -o target\_name file\_name:***Compiles and links* **file\_name** (object files, or generates the object file if .c= file has not been compiled) and generates executable target file with **target\_name** (or a.out by default).
    - **g++** command is a GNU c++ compiler invocation command, which is used for preprocessing, compilation, assembly and linking of source code to generate an executable file. The different “options” of g++ command allow us to stop this process at the intermediate stage.
    - ***g++******file\_name*** command is used to compile and create an executable file *a.out* (default target name).