# CSC 345 - Operating Systems

Project - Virtual OS (minimized)

You may work in groups of three.

---

The goal of this project is to get familiar some ideas of an Operating System. For this project, you will work with a Virtual screen and Virtual keyboard via a custom library. As this OS has little-to-no functionality you will be building drivers (i.e., functions) to it get it working appropriately.

For the first portion of this project, you will work on creating the **keyboard driver**:

- Your driver code should interpret the inputs events (keyboard buttons -> key codes) and map them accordingly.
- Your driver code should interact with the display and input/display the correct keys on screen.

The next portion, will involve taking the input generated by the keyboard and executing it as a command.

Info: The custom library (the Virtual Operating System v1.05.07.2025) will be an early version with-little-to-no other functionality implemented. Only a keyboard and on screen text will be rendered, and it will need the respected drivers created. The next part of the project involves a similar aspect to the **techshell** from the Systems Programming class, which will involve coding the shell portion.

---

The library file contains a few important functions:

- **initializeOS()** - this is used to open a GUI for the operating system (you must register the handlers before you initialize the GUI)
- **registerKeyPressCallback(key_press_callback_t callback)** - this is used to send the event (keycode) to a defined function (callback variable is the function definition)
- **registerCommandHandler(command_handler_callback_t callback)** - this is used to denote which function you use to execute commands.
- **insertKeyData(const char *data)** - sends a character back to the screen (simplified display driver rendering)
- There is also two custom functions you will create to be your main driver functions for the keyboard (callback function) and command executor (also a callback). This function must be of type void and have one parameter of type int.

How to handle the key codes:

- When the keys [**A-Z**], [**0-9**], and [ ] , . / ; ' are pressed they should be displayed on screen.
- When the **Enter** key is pressed it should insert a newline character for now.
- When the **Backspace** key is pressed, send to the insertData function the word **Backspace**
- When the **Esc** key is pressed it should "turn off" the OS by executing the **exit()** function

***Hint: There are a few ways to handle the key codes and driver function: an array, struct, or switch case are recommended methods.*

How to handle executing commands:

- The execution should only happen when you press the enter key (i.e., by passing a newline character to insertKeyData).
- You should use either **popen** or an **exec** variant to execute and return the result to **insertKeyData**.

Your submission should include:

(1) A .c file (your driver code)

(2) A report with a showcase of your keyboard working, commands being executed, and an explanation of the code you implemented