# Lab#3: Moving snake application

Leonardo Fusser (1946995)

## Objectives:

- Code a simple multi-thread application.

**Material:** Windows - Visual Studio

**To hand in:** No report to hand-in. Answers to all questions.

- Indented and commented multi-source file on Git Hub.
- All functions must have a prolog header.
- All files must have their prolog headers: file name, description, date, author and version history.
- Use of symbolic constant is mandatory, all macros in capital letters.
- Answer all questions by filling in the lab sheets using a computer. Submit this document to GitHub.
- Don't erase lines of code of a previous step but comment it out.

You must follow the following encapsulation and modularization rules:

- Abstract all configuration related code by creating functions or macros

```
#define HUMERUS 146.05
void set_arm( float x, float y, float z, float grip_angle_d,)
```

If the requirements are not fulfilled, the student will be asked to re-submit.

**File system structure:**

The file system organization is as listed below. You must populate only the files in **orange.**

- **pdcurses_test**
  - **laboratory**
    - ncurses_init.c
    - **project_files**
      - **mainLab3.c**
      - **threadDisplay**
      - **threadKeybd**
    - **header**
      - public.h
      - ncurses_init.h

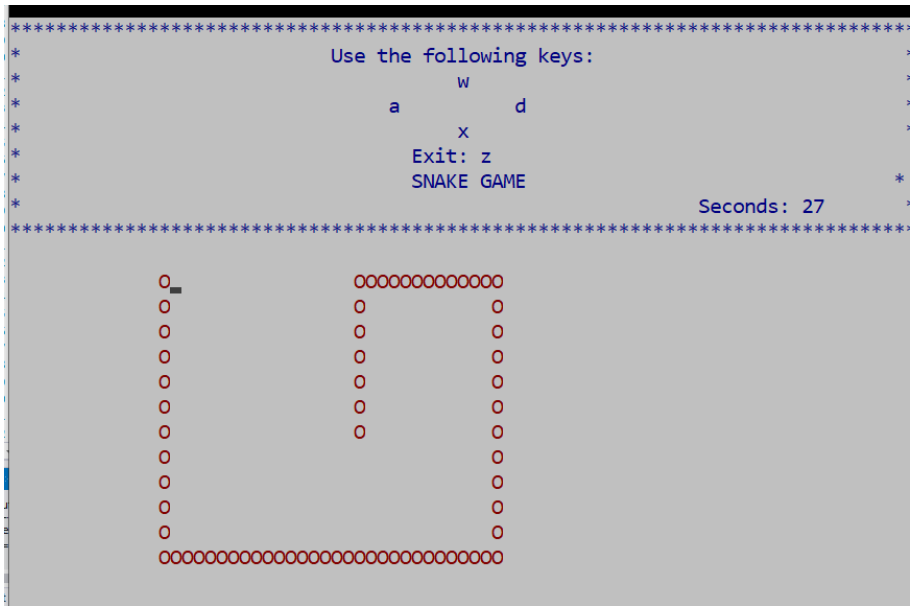
# Lab Work

Clone the previous lab repo:

https://github.com/Embedded-OS-Vanier/lab2-mutex-ncurses-your-name

Requirements:

- You must create a simple moving snake game.
- There should be a menu explaining the usage of the game.
- The snake moves in four directions using 4 keys.
- You game must be split into two threads:
    - a thread to display to the console.
    - a thread that reads the keyboard.
- Both threads are specialized: the keyboard thread must never print to the console and the display thread must never read the keyboard.
- The seconds must be displayed in real time.
- The head of the snake must blink at 2.5 Hz or so.
- The system must never hog the CPU.
- All global variables must be mutex protected.
- The snake must stay within its boundaries. It should never go into the menu area.
- Use the getch() blocking function to read the keyboard.



Since the project is a multi-source file system, you **must** write public functions to create and join. Examples:

```
void createThread1(void)
void pthread_join1(void)
```

Is the system responsive?
If not, you need to tweak your code.

- At first, the system was not responsive. There were two main issues that were encountered: 1) the head of the snake was not being printed properly (head of the snake printing delayed and printing skipped) and 2) the program was hogging the system's CPU.

  The source of the first issue was in the keyboard thread routine. It would be setting x and y values so fast that when the display thread retrieved them (through the use of getters – mutex protected), it would be too slow in doing so. To solve this issue, small blocking delays of around 20mS were used in the keyboard thread routine. This made the snake head correctly print across the entire console window. This also ensured that the system's CPU was not stressed when the keyboard thread routine would run.

  The source of the second issue was in the display thread routine. Although the head of the snake was being printed and moving around correctly across the entire console window, the program would be hogging the system's CPU (shown in Window's task manager). This was because the display thread routine would constantly print the head of the snake to the console window, even when the system was idle. To solve this issue, the code was adapted to be able to tell if the system was idle or not. Additionally, when the program determined that the system is idle, a small blocking delay is used to prevent the system's CPU from being stressed anymore when running the display thread routine.

**Upgrading to your private repository**

Now it is time to update your repo.

Approval before leaving!