121090697 Yang Jiayan Assignment 2 Report

1. Environment

Ubuntu version: 20.04

kernel version: 5.10.197

2. Implementation

(1). `main()`

In `main()`, I initialized the logs' position by using `rand()`. And then I create to threads, one for controlling the logs, and the other for controlling the frog:

```
    pthread_t thread_logs,thread_frog;
    int rc_logs,rc_frog;
    rc_logs=pthread_create(&thread_logs,NULL,logs_move,(void*) 0);
    rc_frog=pthread_create(&thread_frog,NULL,logs_move,(void*) 1);
```

Then I use `pthread_join()` to wait the two processes came to the end. And based on the flag updated during processes' excuting to output the final outcome.

(2). `logs_move()`

As instructed by the explanatory note, I choose to realize the movement of logs and the frog in the same function. So when the input of the function `t==0`, it means we are moving the logs and vice versa.

1). t==0

We first use `usleep(100000)` to control the refresh rate of the movements. Then we check whether the postion of the frog now is in the bound, and if not, we will return immediatly, and set `flag` as 2 (`flag` was initailized as 0 in `main()`, and 1 for winning, 2 for losing, 3 for exiting). Then, we will update the logs' position at the rate 1 char per update.

2). t==1

We need to continue to run the game until we reach the goal (or failed it or exit it). So I used a `while()` loop to check whether `flag` is still 0, which means the game is still running. After every `kbhit()==true`, we will examine the input and to do the corresponding movement. Whether the game is exit or failed is checked during movement, and I check whether the user win the game at the back of the procedure:

```
if(frog.x==0)
{
    flag=1;
    pthread_mutex_unlock(&mutex_frog);
}
```

After that, I print the map again to update the frogs' position on visual.

3. Execution

> Like the instruction in *readme.txt*:

We should first make sure that the path of the terminal is in the files' folder.

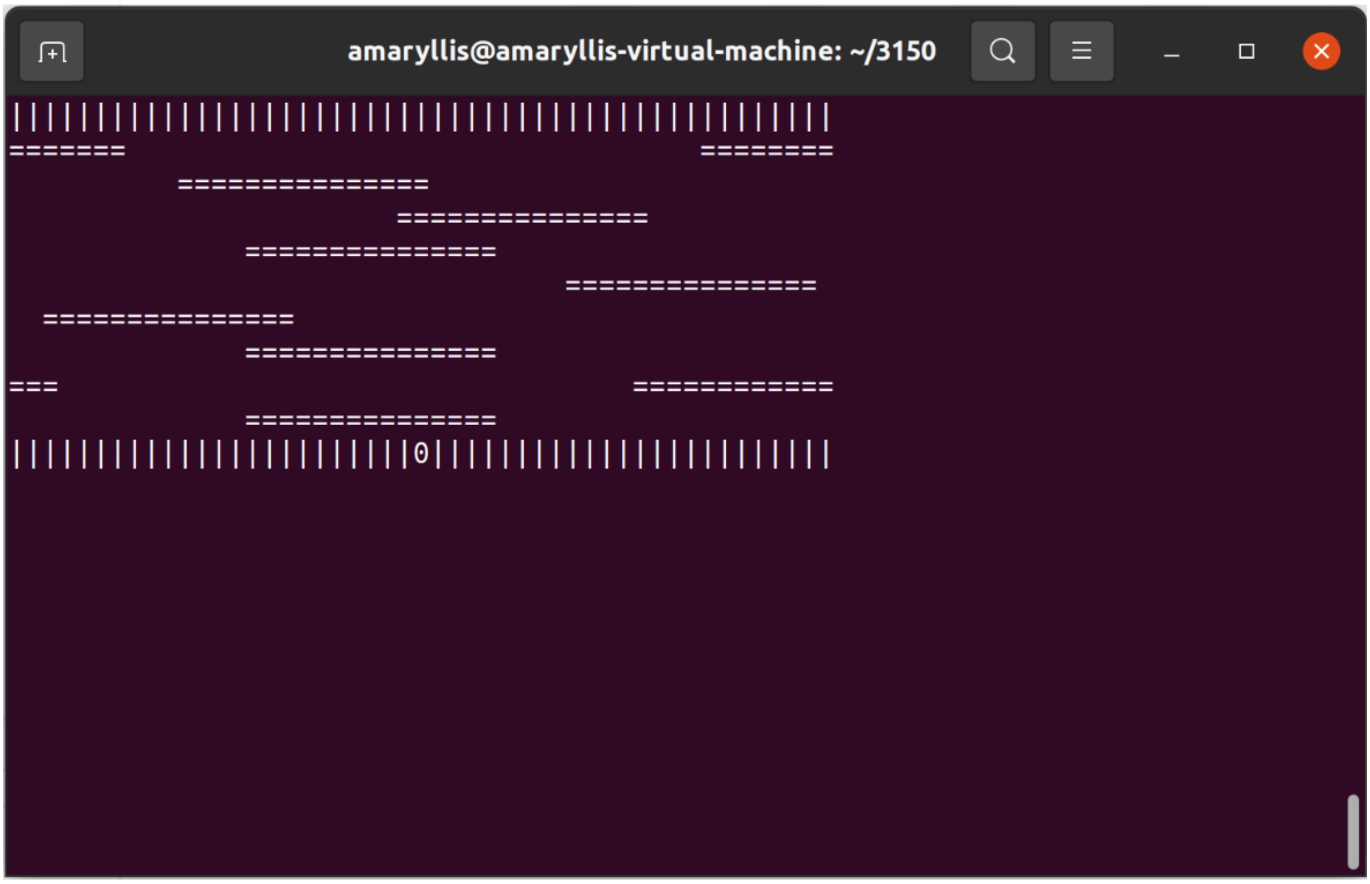
To compile:

```
g++ hw2.cpp -pthread
```

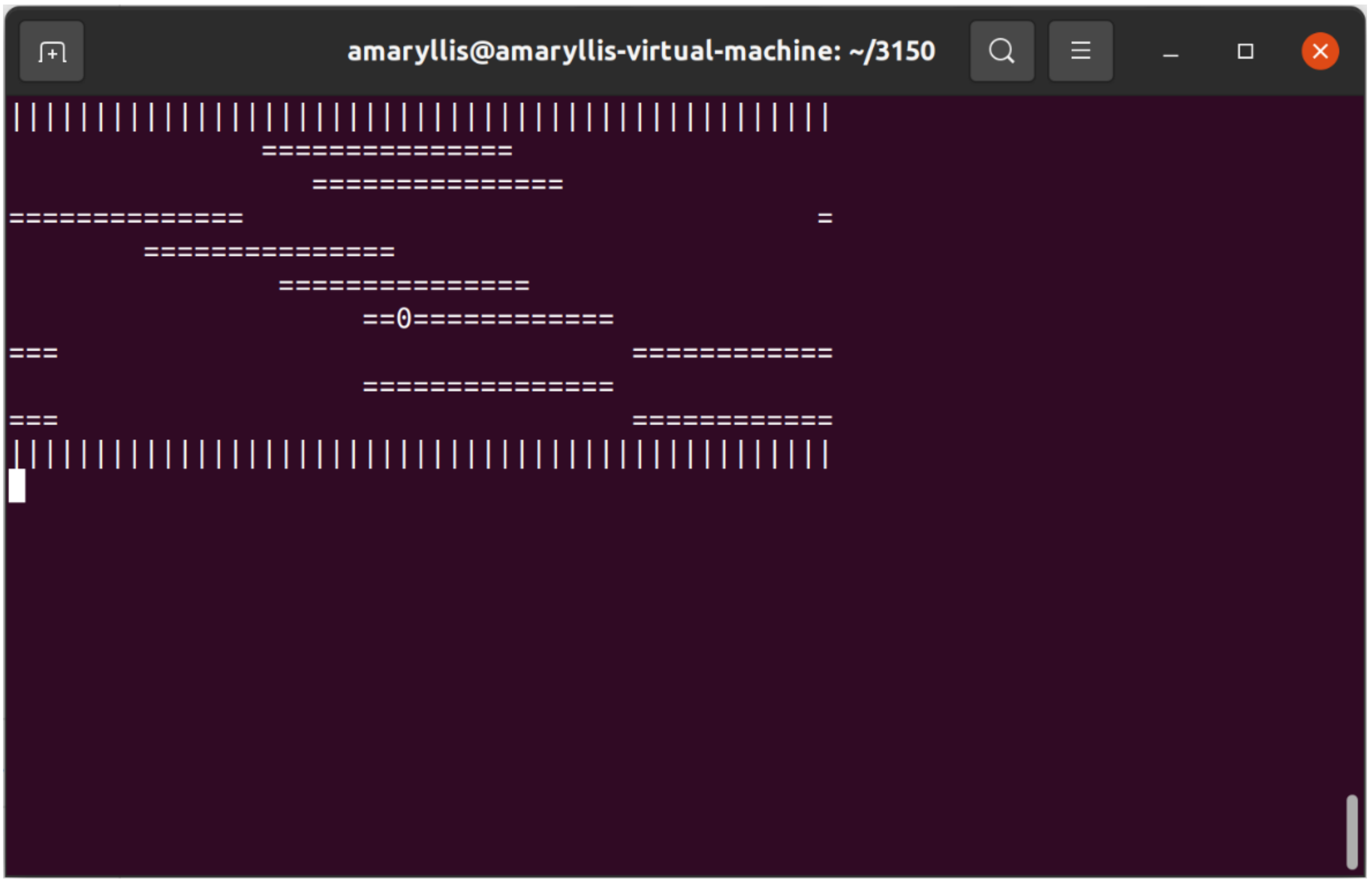
To run:

```
./a.out
```

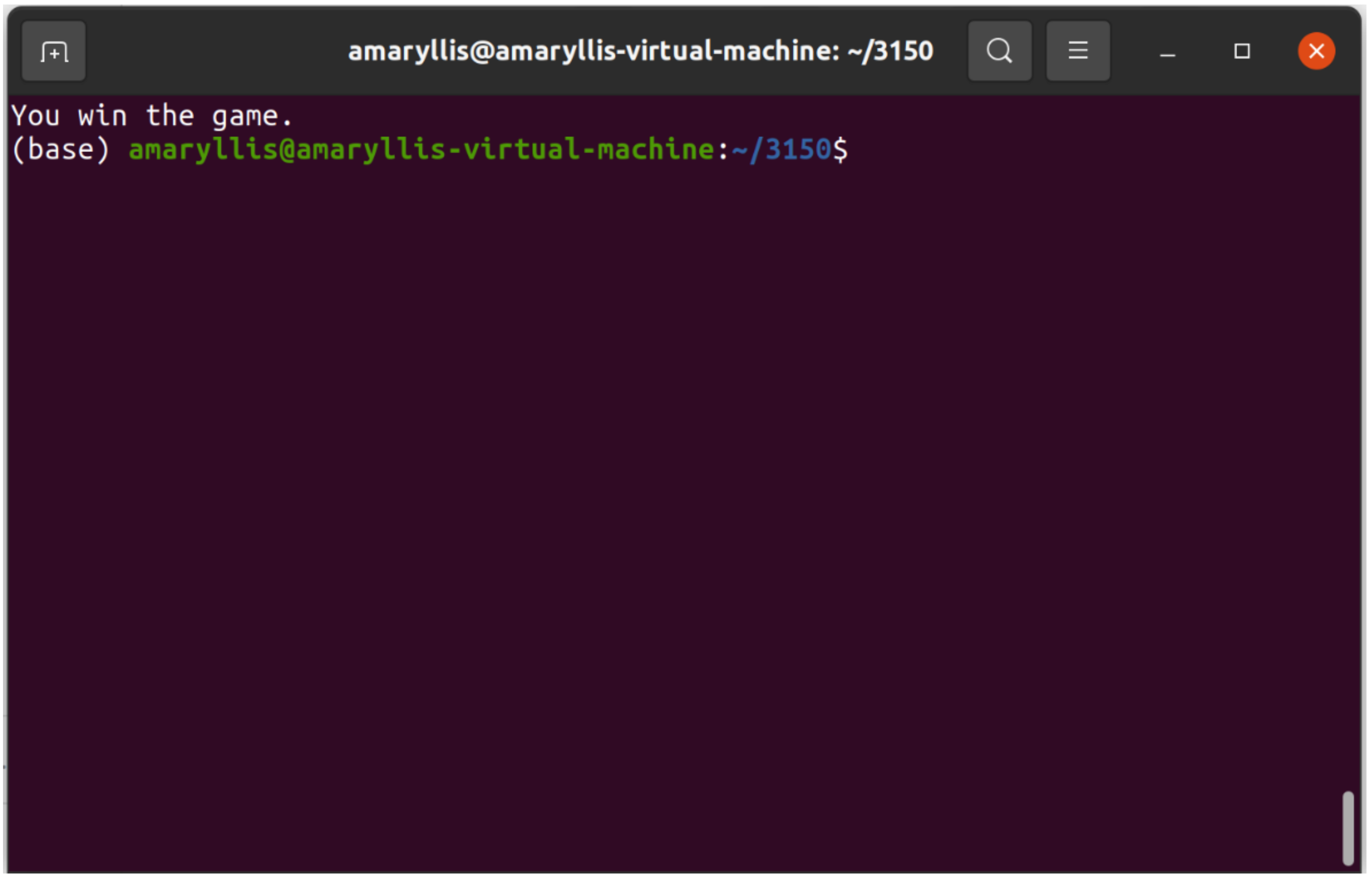
4. Running Screenshots


amaryllis@amaryllis-virtual-machine: ~/3150

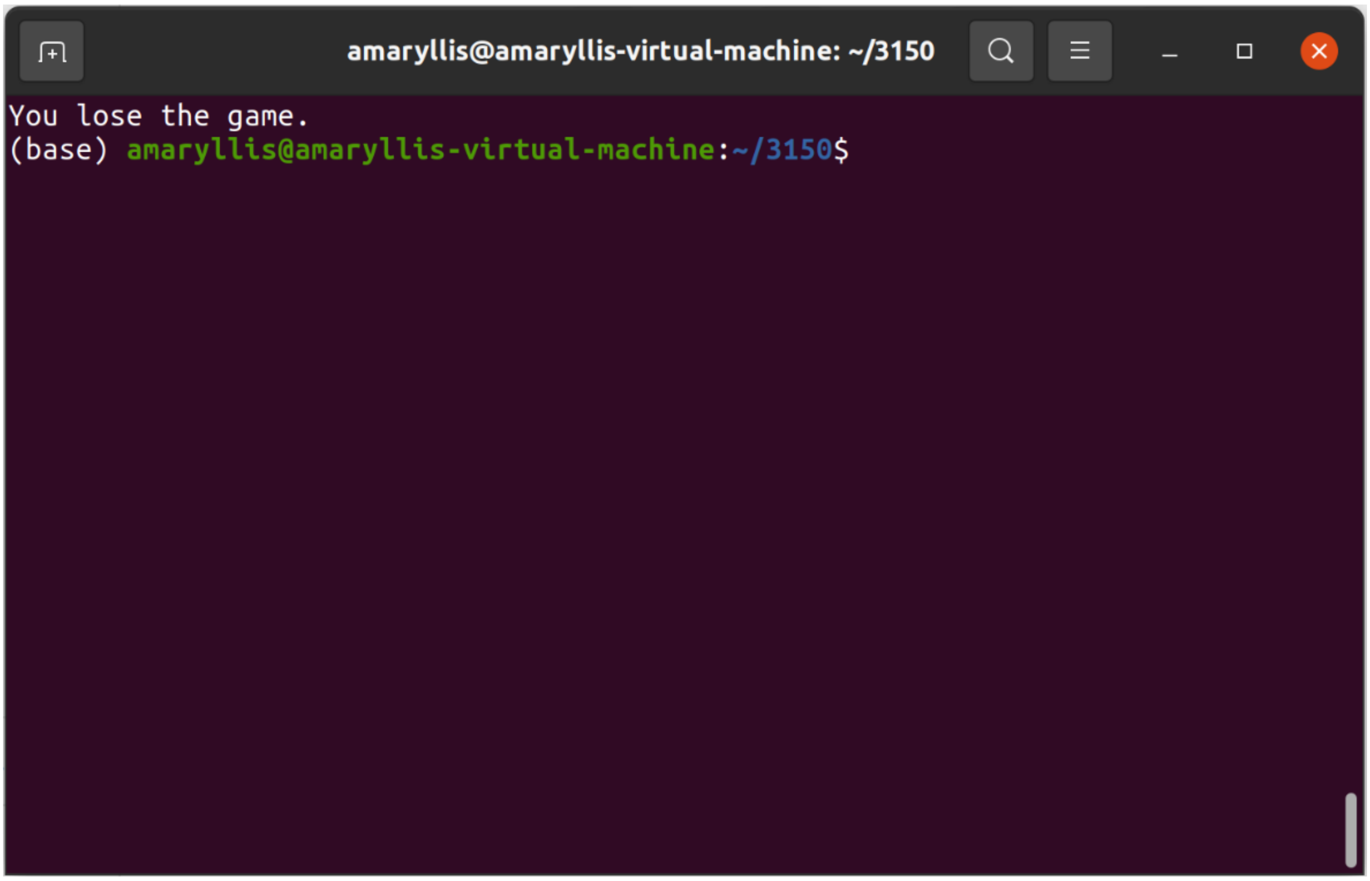
```
|||||||||||||||||||||||||||||||||||||||||||||||||
=======                                  ========
          ===============
                       ===============
              ===============
                                 ===============
  ===============
              ===============
===                                  ============
              ===============
||||||||||||||||||||||||0||||||||||||||||||||||||
```


amaryllis@amaryllis-virtual-machine: ~/3150

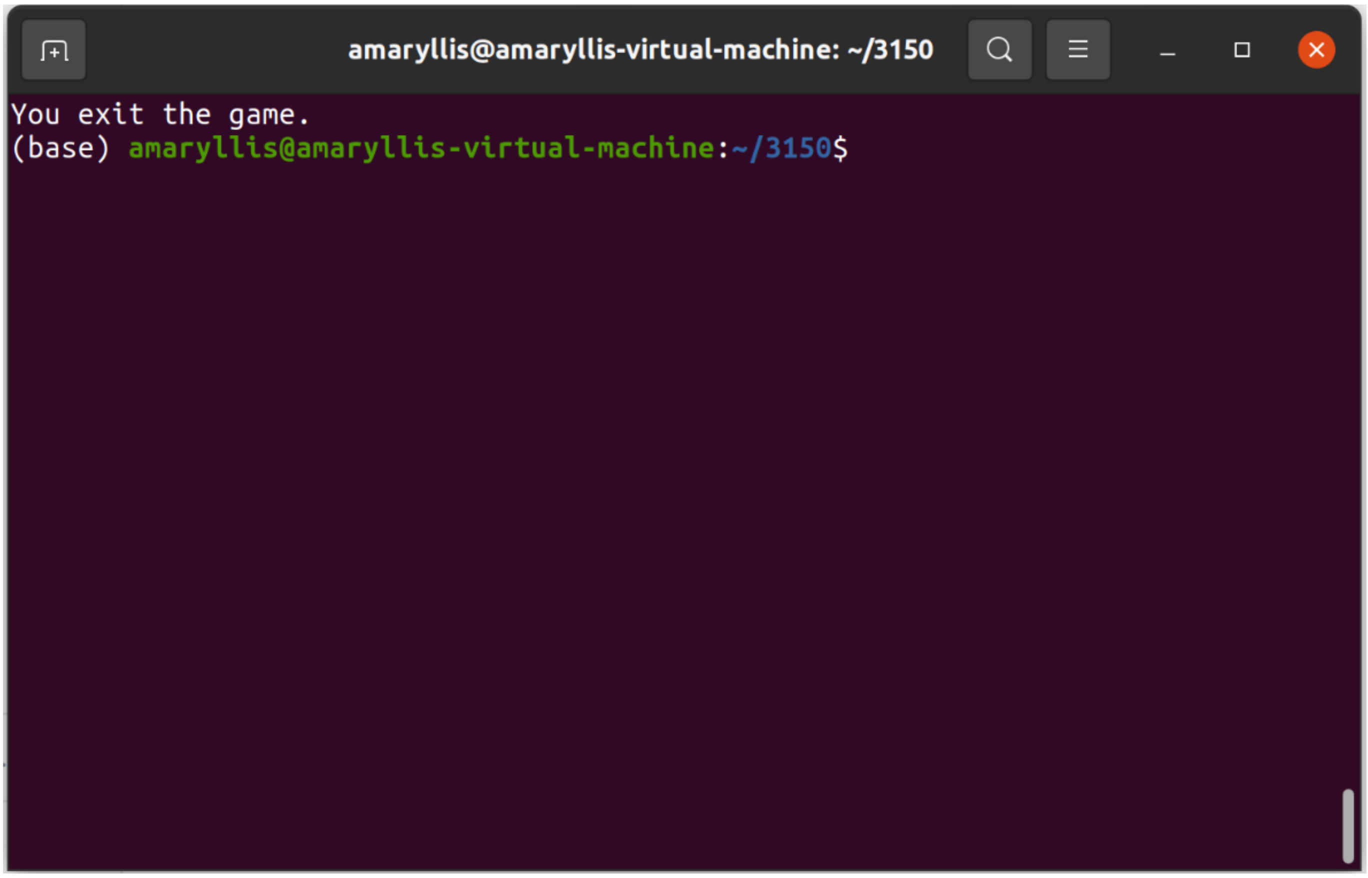
```
|||||||||||||||||||||||||||||||||||||||||||||||||
               ===============
                  ===============
==============                                  =
        ===============
                ===============
                     ==0============
===                                  ============
                     ===============
===                                  ============
|||||||||||||||||||||||||||||||||||||||||||||||||
```



```
You win the game.
(base) amaryllis@amaryllis-virtual-machine:~/3150$
```



```
You lose the game.
(base) amaryllis@amaryllis-virtual-machine:~/3150$
```



```
You exit the game.
(base) amaryllis@amaryllis-virtual-machine:~/3150$
```

5. What I learnt

(1). How to create threads and control them.

(2). How to use mutex.