

3150 Project2

刘芷萱 118010202

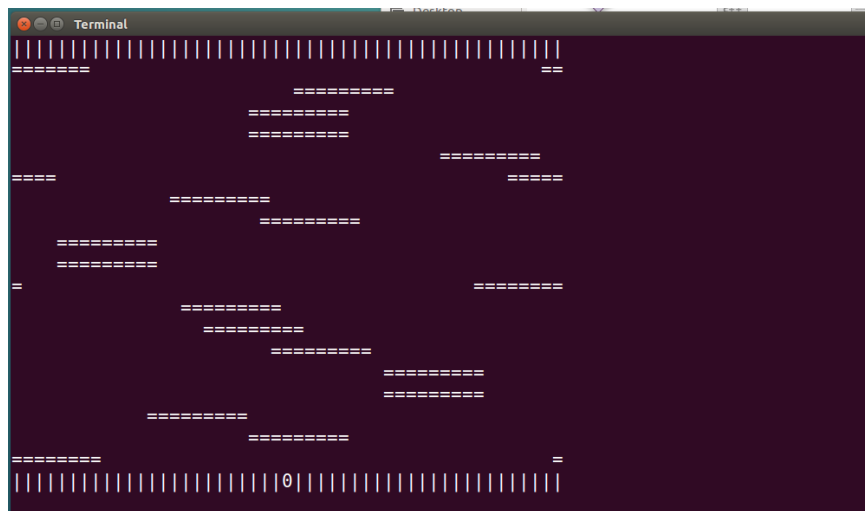
Description of my project:

My environment is Ubuntu 16.04. To compile my program, type 'g++ hw2.cpp -lpthread' and enter on concole in the “source” dictionary. To execute my program, type './a.out' and enter.

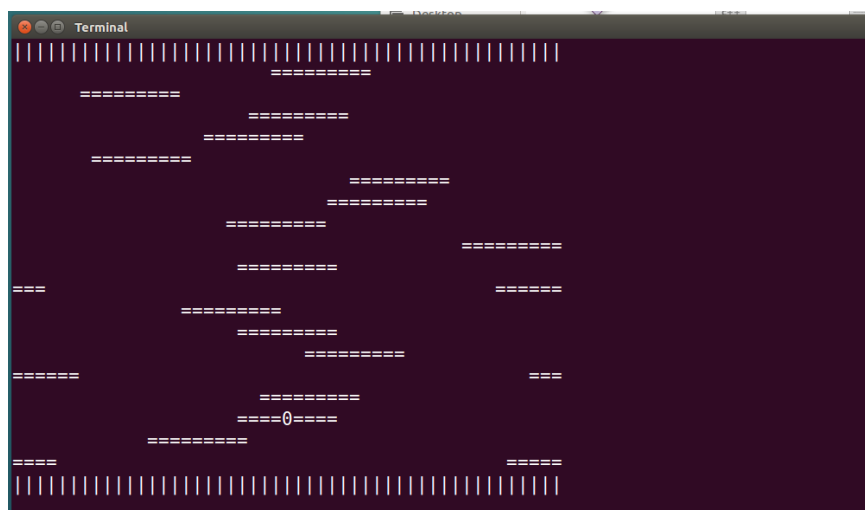
In the first place, you need to choose the mode od difficulty of the game:

```
[10/25/20]seed@VM:~/Downloads$ ./a.out
Enter the mode (easy, normal, hard):
```

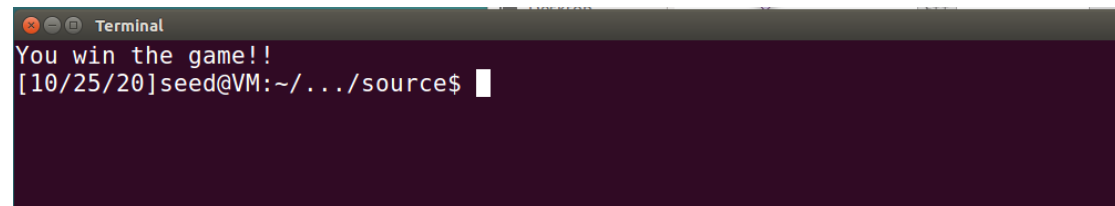
Then, you will enter the game, you will see this figure below, different figure has the different log length and the different speed of log. The figure down below is in the normal mode:



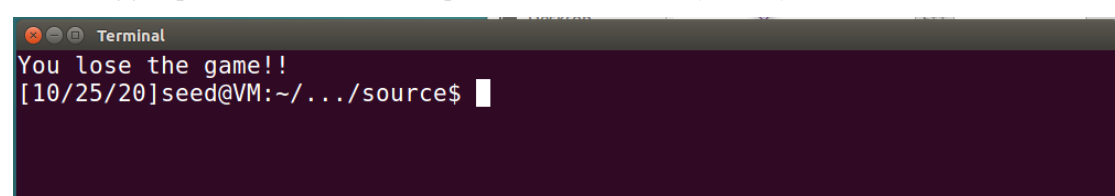
You can use “wads” to control the movement of the “frog” (represented by ‘0’):



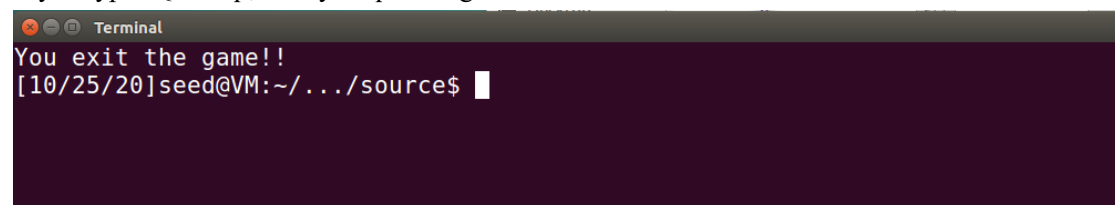
If the frog reach to the bank, you win:

A terminal window with a dark background and light text. The title bar says 'Terminal'. The output shows 'You win the game!!' followed by the prompt '[10/25/20]seed@VM:~/.../source\$' and a cursor.

If the frog jumps into the water or is squeezed to the boundary, then you lose;

A terminal window with a dark background and light text. The title bar says 'Terminal'. The output shows 'You lose the game!!' followed by the prompt '[10/25/20]seed@VM:~/.../source\$' and a cursor.

If you type 'Q' or 'q', then you quit the game:

A terminal window with a dark background and light text. The title bar says 'Terminal'. The output shows 'You exit the game!!' followed by the prompt '[10/25/20]seed@VM:~/.../source\$' and a cursor.

Design of my program

1. In my main function, first I set the boundaries and the frog in a map. Then I use the 'srand' and 'rand' function in my 'log_init' function, in order to randomly set the initial position of each log. I further set the logs in to the map.
2. I create some threads. The number of threads equals to the number of logs plus one frog. For each thread, it will execute the logs_move function.
3. In my 'logs_move' function, I select the threads by its thread number. For threads with odd number, it pushes a certain log to move leftwards. For threads with even number, it pushes other log to move rightwards.
4. For the last threads, it control the frog's movement. I use 'kbhit' function to detect the keyboard input and control the corresponding movement of the frog. In this thread, I also check the frog's status and raise signal. I use the signal to judge whether a game should stop and print out the corresponding prompt.
5. You can see the 'pthread_mutex_lock' in the beginning of my 'logs_move' function, and the 'pthread_mutex_unlock' at the end of this function. This keeps my threads from interfering with each other.

Problem that I met:

I could not quit the game when I detect the keyboard hit. And I found out that this is because only the thread that detect the keyboard hit stopped while others are still running. Then I found out that I could not simply use the 'break' to stop the while loop, since only one thread terminated.

Instead, I set a global variable which is passed by reference. If it is changed no matter by which function, the other functions will detect this change. Finally, my problem solved.

Environment:

As I mentioned before, Ubuntu 16.04.

What I learned:

I learned how to create and terminate pthreads. I learned how to use keyboard control. I learned how to use mutex. I learned how to use the terminal control and so on.....