

CSC3150 A2 Report

HuangPengxiang

10/21/2021

Contents

Overview	3
Some Declarations	3
Enviroment	3
How to Execute My Program	4
The tree of my program	4
The detailed steps of running my program	4
The demo screenshot of my running my program	5
Program Design	5
For the main part	5
For the bonus part	5
Program Implementation	6
Detailed Introduction	6
What I have learned in this prorgam	6
The Problem I met	6

Demo ScreenShot	7
The basic game board	7
The Winning status	7
The losing status	7
The quit status	8

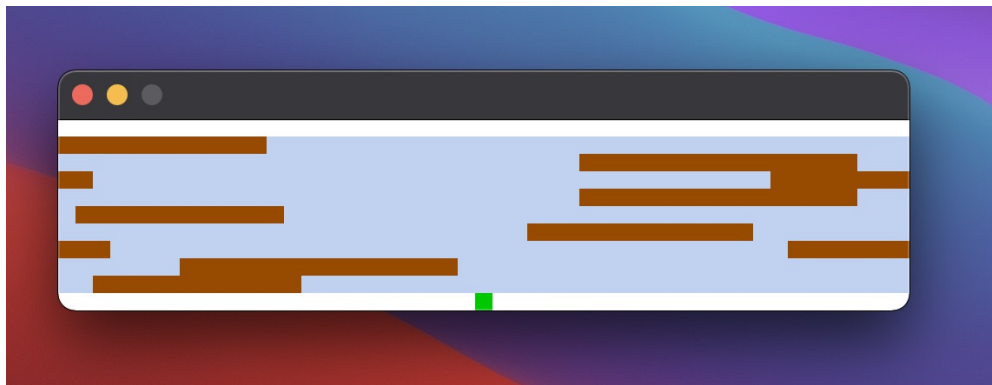
Overview

This is the second Assignment for Operating System. In this assignment, the programming goal is using the multi-thread programming to implement a game named “frog cross river”. The basic rule for this game is that: A river has logs floating on it, and frog must cross the river by jumping on the logs as they pass by. In my program, I implement the whole functionality of this game and also implement **the bonus part** only including generate the random length of the log.

Some Declarations

There might be some Indentation program in your test computer, since I write my program in my mac, and there are some in incompatibility in your test computer. Although I adjust it in my Linux system, I am not sure it may appear some odd indentation in your computer. For The bonus part, I use QT implement the GUI interface, however, when I try to implement the multithread version, there are always some odd bugs appear, whenever I move the log and frog concurrently, it crashed. I spend a long time but can not fix it till deadline. And I hangout the incomplete version of my qt program if you would like to see.

some demo picture for GUI part:



```
11:29:04: Starting /Users/huangpengxiang/Desktop/build-gui3150qt-Desktop_Qt_5_12_10_clang_64bit-Debug/gui3150qt.app/Contents/MacOS/gui3150qt ...
11:29:14: 程序异常结束。
11:29:14: The process was ended forcefully.
11:29:14: /Users/huangpengxiang/Desktop/build-gui3150qt-Desktop_Qt_5_12_10_clang_64bit-Debug/gui3150qt.app/Contents/MacOS/gui3150qt crashed.
```

Enviroment

The running environment is showing below:

- The linux version:

```
huangpengxiang@ubuntu:~$ uname -r
4.15.0-142-generic
huangpengxiang@ubuntu:~$
```

- The gcc version:

```
huangpengxiang@ubuntu: ~
huangpengxiang@ubuntu:~$ gcc --version
gcc (Ubuntu 5.4.0-6ubuntu1~16.04.12) 5.4.0 20160609
Copyright (C) 2015 Free Software Foundation, Inc.
This is free software; see the source for copying conditions. There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
```

How to Execute My Program

The tree of my program

```
huangpengxiang@ubuntu:~/Desktop/3150_Project$ tree
.
├── Report.pdf
├── source
│   ├── hw2.cpp
│   ├── makefile
│   └── README.txt
└── 1 directory, 4 files
```

The detailed steps of running my program

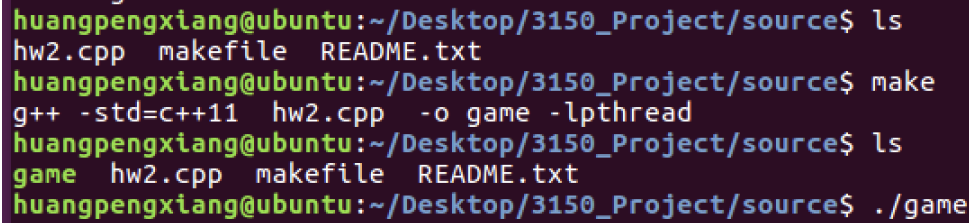
```
$ cd /* the source directory located */
```

```
$ make          # I write a makefile to link the library where pthread located
```

```
$ ./game        # the execute file is named 'game'
```

```
$ make clean    #remember to clean
```

The demo screenshot of my running my program



```
huangpengxiang@ubuntu:~/Desktop/3150_Project/source$ ls
hw2.cpp  makefile  README.txt
huangpengxiang@ubuntu:~/Desktop/3150_Project/source$ make
g++ -std=c++11 hw2.cpp -o game -lpthread
huangpengxiang@ubuntu:~/Desktop/3150_Project/source$ ls
game  hw2.cpp  makefile  README.txt
huangpengxiang@ubuntu:~/Desktop/3150_Project/source$ ./game
```

Program Design

For the main part

For the main part, I divide the program into 4 parts including initialization, log move, frog move, and result printing. **For the initialization**, I implement the location and the length for each log. After initialization, the frog will in the middle location of the base side river, and each log will randomly appear at each location with different length. **For the log_move**, I used 9 thread to control and each thread control one log, and each thread will move one by one will very small latency, which human can not see and just judge them move synchronously, and each log will come back to another side and move across the screen after it touch one side of screen. Moreover, the odd log will move left and even log move right in case to make sure user have the possibility to win the game. **For the frog move**, frog is the number 10 thread, which means frog will wait for the log move and move by user command. **For the result part**, I set a ISOVER variable to check whether the game is over or user just quit the game, and clear the terminal and print out the corresponding result.

For the bonus part

For the bonus part, **I only implement the random length of the log in the initialization part**. For the GUI part, I try to use SDL2, Imgui, OpenGL to implement the graphic output. however, The time is limited and I didn't learn the basic information about the GUI design, and graphic developing in C++ is kind of difficult. So I didn't finish the GUI part but it is interesting to design a graphic interface for user in this game, maybe I will try this one later.

Program Implementation

Detailed Introduction

For the Initialization part, I use *rand()* function in random library to randomly generate the length number between some range. For the log move part, I use *pthread_create()* to create 9 threads of log, and for each log, it will have the unique id stored in my global variable. each thread will jump into the *mutex()* one by one, and execute it, which is move one step in the screen. and also I set a latency, which is function *usleep()* to avoid thread have the confliction and make each thread move one by one with very small latency. And I also use *pthread_cond_signal* to send the signal to frog when every thread already moved. After log move, the thread of frog will execute the function *frog_move()*. frog will basically move due to the user input command, user can only use W, A, S, D to control the direction and can input q to quit the game. I also set a status variable to check the game status, if the status return 1 or -1, means game is over, and 0 means game continue, the frog thread will go to the end of thread queue and the first log will continue to execute it until the game is over. Finally, If the game is over, The program will clean the screen and print out the result.

What I have learned in this program

The most important knowledge I learned is thread control, which includes thread creation thread allocation, and how to use lock to avoid dead lock. I also learned how to divide a whole game into different part and thread, and implement them thread by thread. The communication between thread is also important, I use *thread_cond_signal* to implement it, which is set a global condition variable, when one thread satisfy the global condition, it will send the signal to another thread, let another thread to execute and wait until it end. Since there are multiple threads which will execute the function, we need to ensure that there is only one thread executing the function at the time.

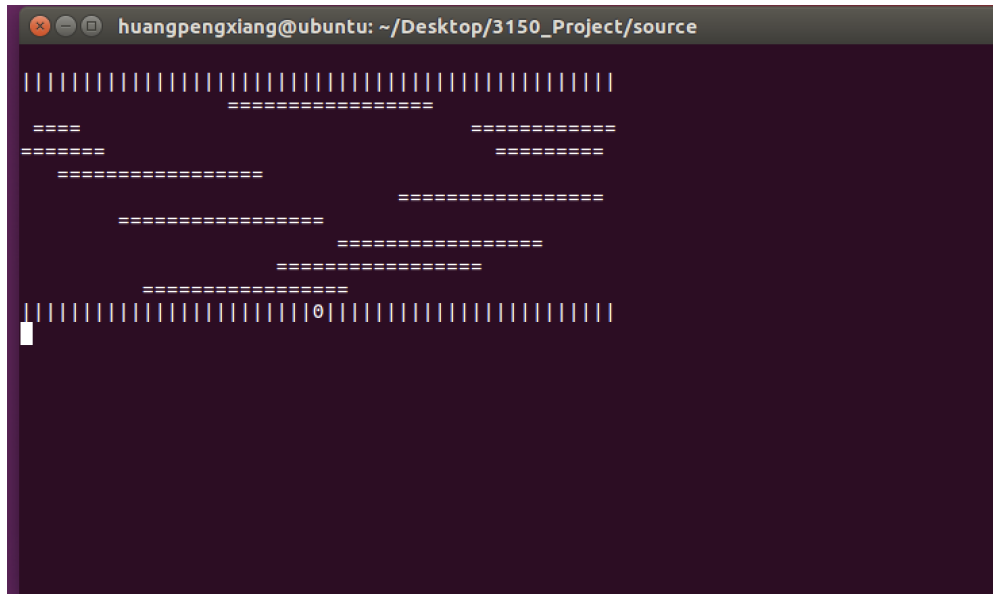
The Problem I met

When I set the latency too small, since I want to let the game be fast. But the screen will stuck. I think maybe the very small latency is not ok since another thread will come into the while loop and they both want to the mutex and ten thread will have the confliction for the mutex. So I set a higher latency to make sure there are only thread is execute in the lock every time and keep them in order.

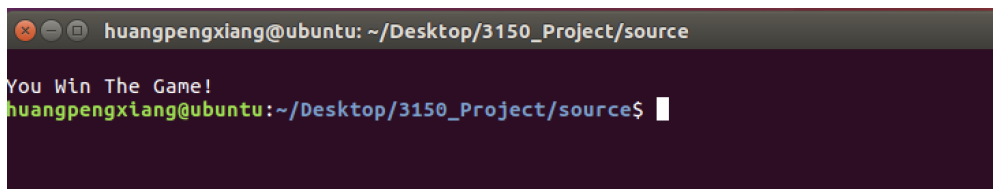
I initially put the wait signal function in the middle of the move log loop, only before puts function. The game will occasionally exit unexpectedly since the map is manipulated by the two threads simultaneously. The solution is put the wait signal function at the beginning of the move log function.

Demo ScreenShot

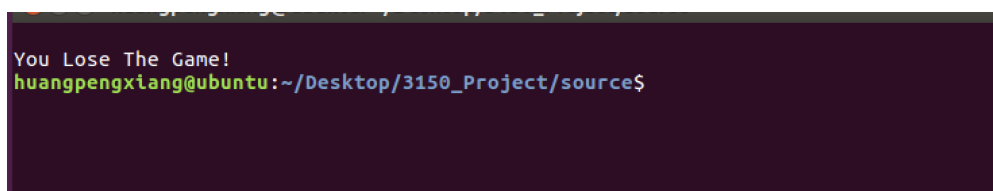
The basic game board



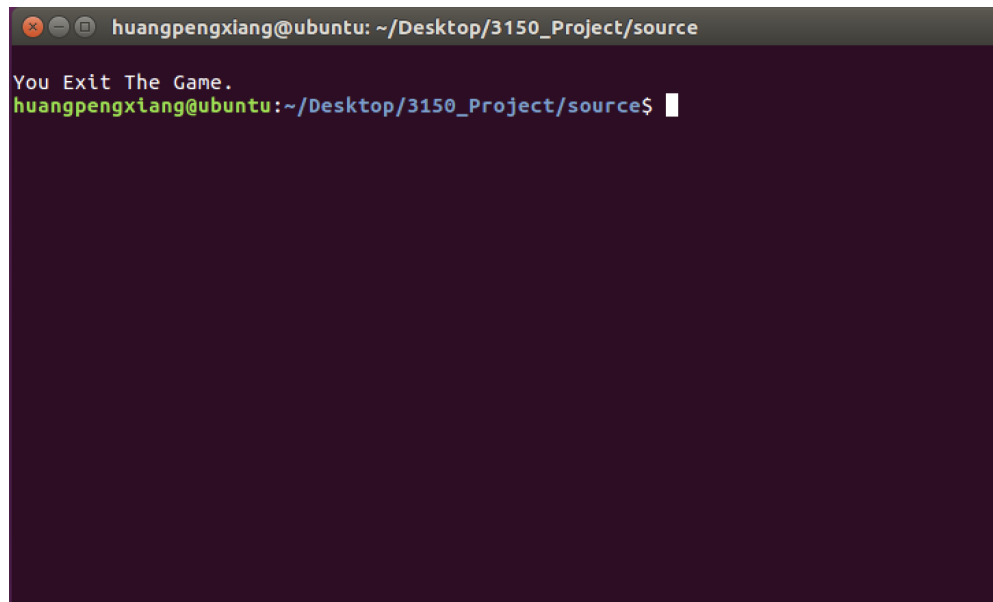
The Winning status



The losing status



The quit status



```
huangpengxiang@ubuntu: ~/Desktop/3150_Project/source
You Exit The Game.
huangpengxiang@ubuntu:~/Desktop/3150_Project/source$
```

A terminal window with a dark purple background. The title bar at the top shows window control icons and the text "huangpengxiang@ubuntu: ~/Desktop/3150_Project/source". The terminal content shows the message "You Exit The Game." followed by the prompt "huangpengxiang@ubuntu:~/Desktop/3150_Project/source\$" and a cursor.