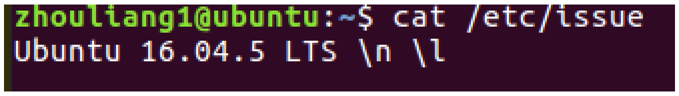
CSC3150 Assignment2

Yu, Zhouliang 120040077

October 2021

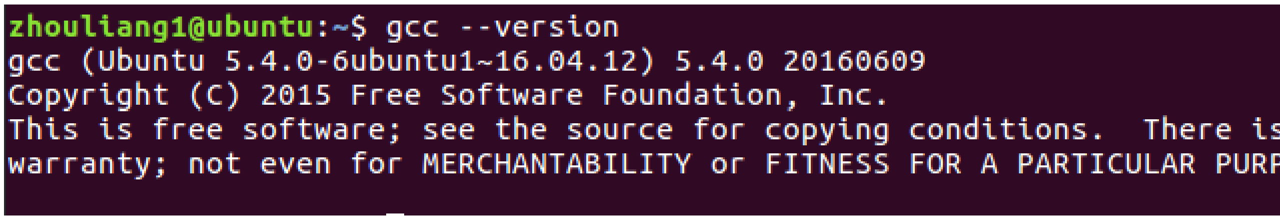
1 Environment

1.1 Linux Version



```
zhouliang1@ubuntu:~$ cat /etc/issue
Ubuntu 16.04.5 LTS \n \l
```

1.2 GCC Version



```
zhouliang1@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
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURP
```

2 The Step to Execute The Program

2.1 Compile

In the 'source' directory, type 'g++ hw2.cpp -lpthread' and enter on concole.

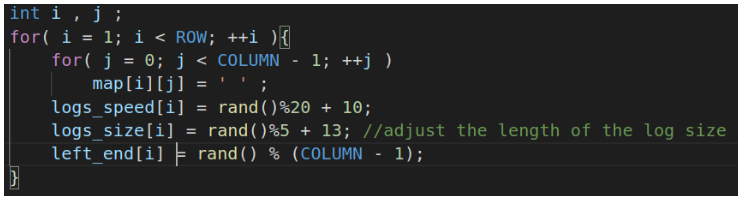
2.2 Execute

In the 'source' directory, type './a.out'

3 The Design of My Program

3.1 logmove desgin

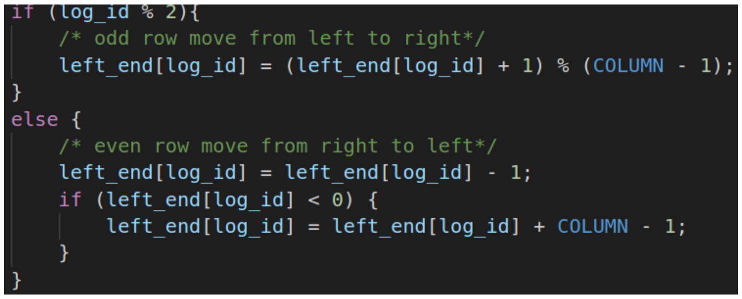
first set the property of the logs to make their speed vary each other, randomly generated length, and randomly generated leftend



```
int i , j ;
for( i = 1; i < ROW; ++i ){
    for( j = 0; j < COLUMN - 1; ++j )
        map[i][j] = ' ' ;
    logs_speed[i] = rand()%20 + 10;
    logs_size[i] = rand()%5 + 13; //adjust the length of the log size
    left_end[i] = rand() % (COLUMN - 1);
}
```

in order for some log to move from left to right and some from right to left, I set the log on the odd row moving from left to right, while log on the even row moving from right to left

the motion of move is implemented by the operation on the array to load the left end of each log, and use mod to allow the log to go across the boundary and emerging at the other side.



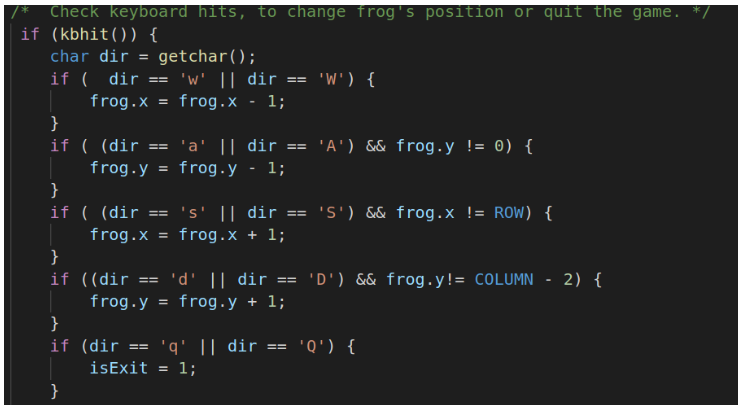
```
if (log_id % 2){
    /* odd row move from left to right*/
    left_end[log_id] = (left_end[log_id] + 1) % (COLUMN - 1);
}
else {
    /* even row move from right to left*/
    left_end[log_id] = left_end[log_id] - 1;
    if (left_end[log_id] < 0) {
        left_end[log_id] = left_end[log_id] + COLUMN - 1;
    }
}
```

3.2 create pthread and use mutex for controling frog and logs

1. we create pthreads for each log and the downside land, their executable routine are all log_move, to enable them to control the motion of each logs and plot maps, with arg to be '(void*) t' which can transfer to log_id
2. each pthread_create is paired with pthread_exit if main finishes before the threads it has created, and exit with pthread_exit the other thread will continue to execute. otherwise, they will all automatically terminated when main finishes. so we set a pthread_exit at the end of main and end of log_move
3. to make the multi-thread process accomplish synchronization between threads we use pthread_join to subroutine blocks the calling thread until the specific thread terminated
4. in order for protecting the sharing data in each thread preventing other thread to write in that thread, we use mutex to lock the pthread while its functioning

3.3 keyboard actions capture

in each pthread we are going to check if there is a keyboard hit, use 'getchar' to capture the action if we press w the frog move upward, s to move downward, d to move right, a to move left, if we press q we quit

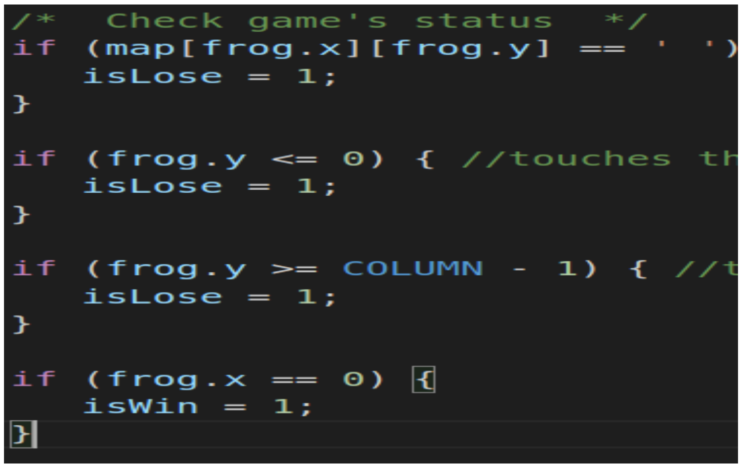


```
/*  Check keyboard hits, to change frog's position or quit the game. */
 if (kbhit()) {
    char dir = getchar();
    if (  dir == 'w' || dir == 'W') {
        frog.x = frog.x - 1;
    }
    if ( (dir == 'a' || dir == 'A') && frog.y != 0) {
        frog.y = frog.y - 1;
    }
    if ( (dir == 's' || dir == 'S') && frog.x != ROW) {
        frog.x = frog.x + 1;
    }
    if ((dir == 'd' || dir == 'D') && frog.y!= COLUMN - 2) {
        frog.y = frog.y + 1;
    }
    if (dir == 'q' || dir == 'Q') {
        isExit = 1;
    }
```

3.4 game status judging and message print

we set booleans 'isWin', 'isLose','isExit' to denote the status, and in the function log_move we are able to judge the status in each thread. if frog falls to the rive or touches the boundary left and right, isLose will be set 1, if it goes to the other side, isWin will be set to one, if we quit, isExit will be set to 1

and if any of these status changes, we are able to print the corresponding message to the user



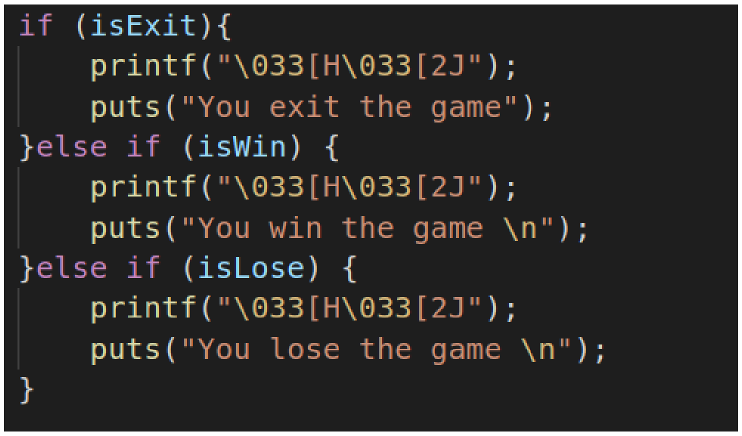
```
/*  Check game's status  */
if  (map[frog.x][frog.y]  ==  '  ')
    isLose = 1;
}

if  (frog.y <= 0) { //touches th
    isLose = 1;
}

if  (frog.y >= COLUMN - 1) { //t
    isLose = 1;
}

if  (frog.x == 0) {
    isWin = 1;
}
```

the print out message is

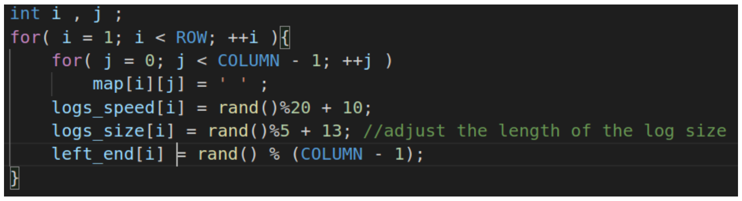


```
if (isExit){
    printf("\033[H\033[2J");
    puts("You exit the game");
}else if (isWin) {
    printf("\033[H\033[2J");
    puts("You win the game \n");
}else if (isLose) {
    printf("\033[H\033[2J");
    puts("You lose the game \n");
}
```

4 bonus

4.1 length of logs random generating

we use srand and random function to generate random length logs

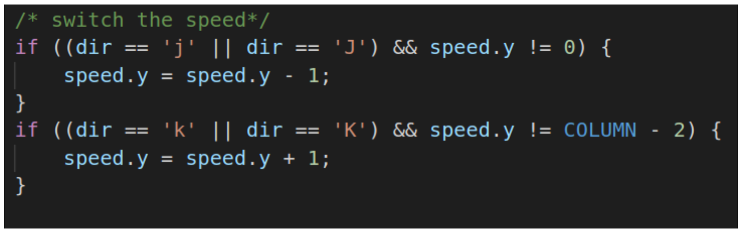


```
int i , j ;
for( i = 1; i < ROW; ++i ){
    for( j = 0; j < COLUMN - 1; ++j )
        map[i][j] = ' ' ;
    logs_speed[i] = rand()%20 + 10;
    logs_size[i] = rand()%5 + 13; //adjust the length of the log size
    left_end[i] = rand() % (COLUMN - 1);
}
```

4.2 slide bar to adjust speed

in my program I define a new object speed, the speed of logs can be adjust by the position of speed symbol

in my program, we can press 'K' or 'k' to increase the speed, and press 'j' or 'J' to decrease the speed



```
/* switch the speed*/
if ((dir == 'j' || dir == 'J') && speed.y != 0) {
    speed.y = speed.y - 1;
}
if ((dir == 'k' || dir == 'K') && speed.y != COLUMN - 2) {
    speed.y = speed.y + 1;
}
```

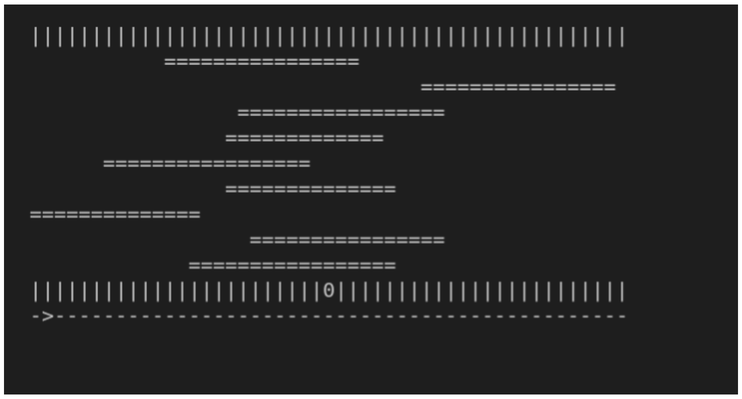
4.3 GUI

This can be implemented by outer graphical library

5 Sample Output

5.1 interface

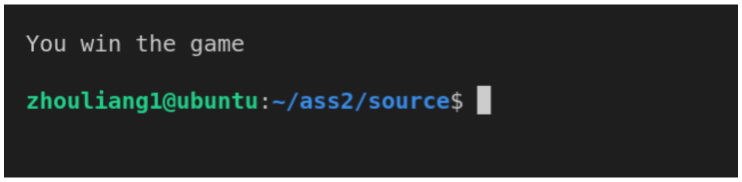
the interface of game, you will find the length of each logs is not different, because they are random generated



```
|||||||||||||||||||||||||||||||||||||||||||||||||
           ================
                                ================
                 =================
                =============
      =================
                ==============
==============
                  ================
             =================
||||||||||||||||||||||||0||||||||||||||||||||||||
->-----------------------------------------------
```

5.2 win

you win the game

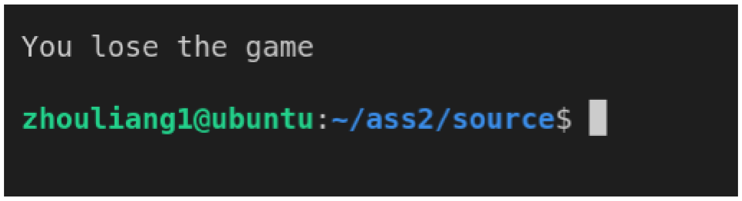


```
You win the game

zhouliang1@ubuntu:~/ass2/source$ 
```

5.3 lost

you lose the game

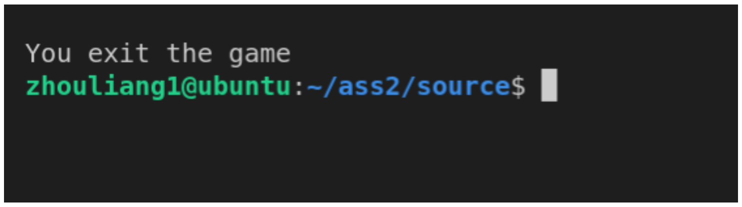


```
You lose the game

zhouliang1@ubuntu:~/ass2/source$ 
```

5.4 quit

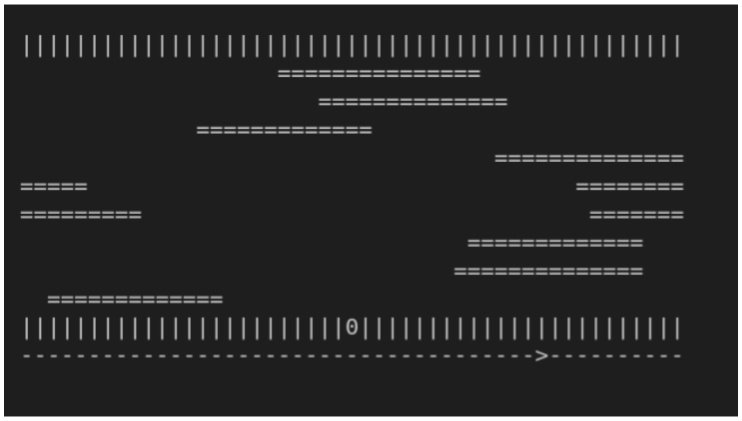
quit the game



```
You exit the game
zhouliang1@ubuntu:~/ass2/source$ 
```

5.5 speed

adjust the speed, k for right, j for left



```
|||||||||||||||||||||||||||||||||||||||||||||||||
                   ===============
                      ==============
             =============
                                   ==============
=====                                    ========
=========                                 =======
                                 =============
                                ==============
  =============
||||||||||||||||||||||||0||||||||||||||||||||||||
-------------------------------------->----------
```

6 What I Learnt From the task

6.1 -lpthread

Q: Why gcc does not link to the pthread by "#include<pthread.h>", while we must use "-lpthread" to compile the program
A:

Having #include <pthread.h> in your code doesn't link in the library; it only includes the header for compilation. That allows the compiler to see the various structures, function declarations, etc. included. Having -lpthread actually causes the linking to be done by the linker. So the include tells the compiler what's available, and the -lpthread actually allows the program to call the functions within the library at runtime.

6.2 pthread_join

Q: Why do we use pthread_join in multiple thread programming
A: if your main thread(the thread executed by the main), exits before other threads, then it will cause bugs. With pthread_join, the main thread is waiting until other thread exit.

6.3 pthread_mutex

Q: Why do we need a mutex?
A: A mutex is a mutually exclusive flag, it acts as a gate keeper to a cection of code allowing one thread in and blocking access to al others, this ensures that code being controled will only be hit by a single thread each time, be sure to release the mutex when you are done.

reference: https://stackoverflow.com/questions/34524/what-is-a-mutex