Design: I finished the program in a design that log should be managed by its own thread, so I design a log\_move thread function and create thread for each log with pthread library. The thread function will decide if the log should move leftwards or rightwards. And everytime it get scheduled, it update the log’s position and handle keyboard interrupt if encounters. If no keyboard hit occurs, the thread will just move its floating log and go to sleep for a short moment, meanwhile another thread will be scheduled. Since each thread just execute for a split second, we can hardly distinguish the order of log movement. We just feel that all the logs move simultaneously. Our keyboard interrupt will be handled by the thread just on scheduled.

Mutex employment:

When thread function handles keyboard interrupt and update log map, the thread will request a mutex, and after it finished its work this time, the mutex will be released and another thread blocked before can then go on.This mechanism ensures that no disorder will be introduced when multithread is employed.

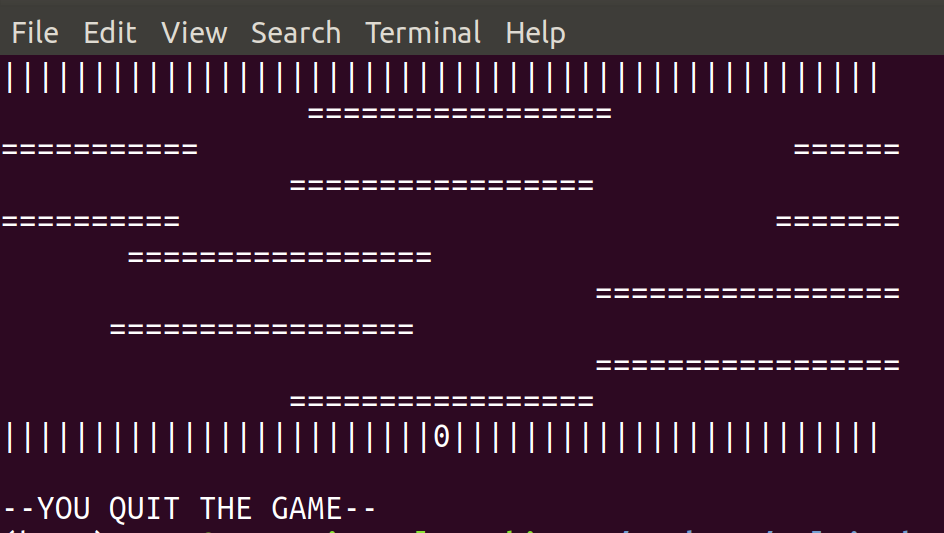
Problem encountered:

Disorder occurred when mutex was not employed at first.

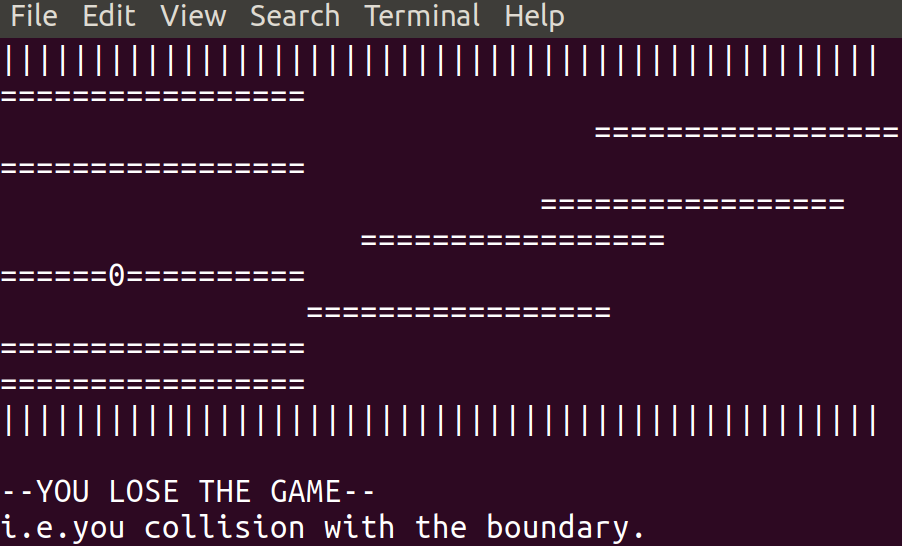
The steps to execute:

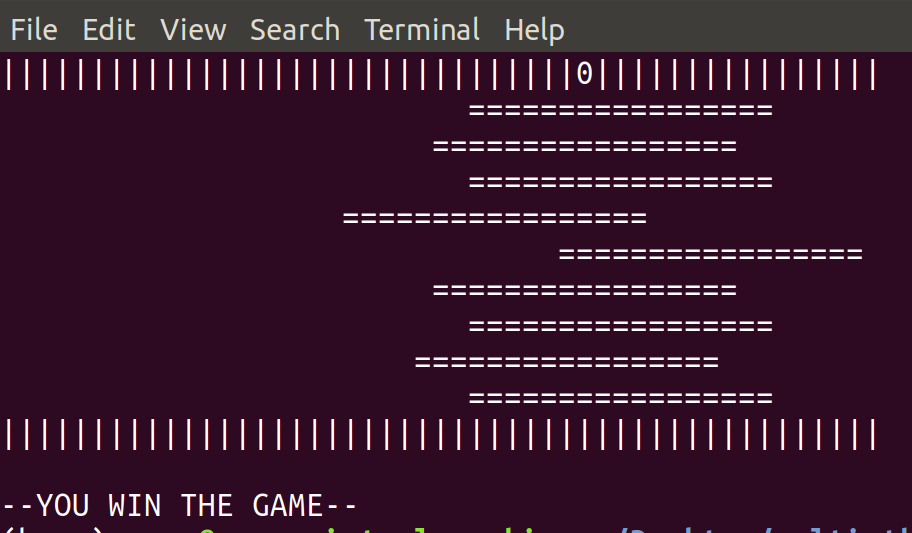
1. Compile and link with pthread g++ -o main main.cpp -lpthread
2. Execute from terminal ./main

Screenshot of program output:









Learn from this assignment:

Design mode is important, a good design will reduce workload and get more qualified and robust program. To get a good design, we must have a sharp insight to the problem and make a good abstract