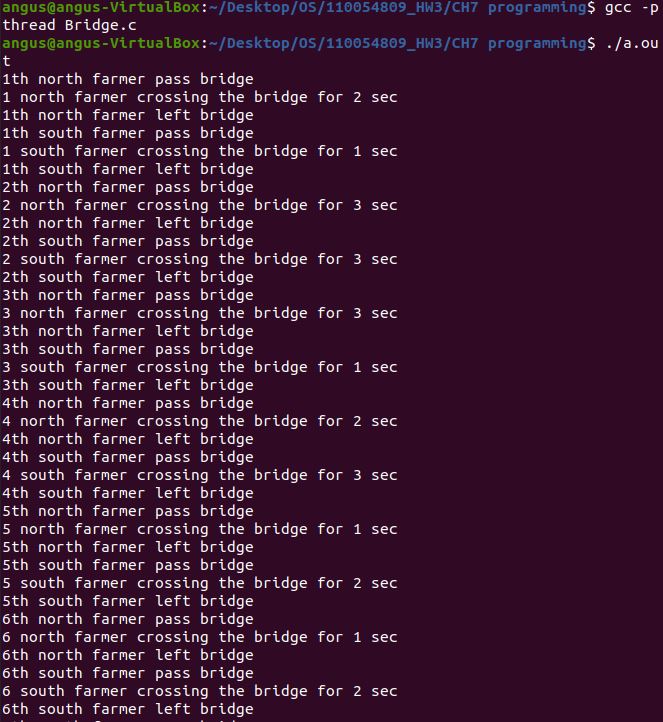
Steps：

1. gcc –pthread Bridge.c
2. ./a.out



Source code：

#include<stdio.h>

#include<stdlib.h>

#include<pthread.h>

#include<unistd.h>

#include<time.h>

#define threads 10

int north = 1, south = 1;

pthread\_mutex\_t mutex = PTHREAD\_MUTEX\_INITIALIZER;

void bridge(char from[], int index) {

srand(time(NULL));

int crossTime = rand() % 3 + 1;

printf("%d %s farmer crossing the bridge for %d sec\n", index, from, crossTime);

sleep(crossTime);

}

void \*northFarmer() {

pthread\_mutex\_lock(&mutex);

printf("%dth north farmer pass bridge\n", north);

bridge("north", north);

printf("%dth north farmer left bridge\n", north);

north++;

pthread\_mutex\_unlock(&mutex);

pthread\_exit(0);

}

void \*southFarmer() {

pthread\_mutex\_lock(&mutex);

printf("%dth south farmer pass bridge\n", south);

bridge("south", south);

printf("%dth south farmer left bridge\n", south);

south++;

pthread\_mutex\_unlock(&mutex);

pthread\_exit(0);

}

int main() {

pthread\_t north[threads], south[threads];

pthread\_mutex\_init(&mutex, NULL);

for (size\_t i = 0; i < threads; i++) {

pthread\_create(&north[i], NULL, northFarmer, NULL);

pthread\_create(&south[i], NULL, southFarmer, NULL);

}

for (size\_t i = 0; i < threads; i++) {

pthread\_join(north[i], NULL);

pthread\_join(south[i], NULL);

}

pthread\_mutex\_destroy(&mutex);

return 0;

}