Lab 2 Report

CIS 657 Principles of Operating Systems

Name: Xiaoqian Huang

SUID: 878174727

CIS657 Fall 2019
Assignment Disclosure Form

Assignment #: Lab 2

Name: Xiaoqian Huang

1. Did you consult with anyone other than instructor or TA/grader on parts of this assignment?
   If Yes, please give the details.
   No.

2. Did you consult an outside source such as an Internet forum or a book on parts of this assignment?
   If Yes, please give the details.
   No.

I assert that, to the best of my knowledge, the information on this sheet is true.

Signature:_____Xiaoqian Huang_______________ Date : 2019/9/13

1. **Time Consuming**
   1) Analyze the problem, determine specifications, create the design: 30 minutes.
   2) Implement the design: 4 hours (first version of implementation).
   3) Test/debug the program: 18 hours (make the program compiled and run, fix logical errors, fix segmentation fault errors, print and analyze large amount of output data to debug).
2. **Design/Solution for each requirement**

   First of all, I created a simulator which is declared at Simulator.h and defined at Simulator.cc to simulate the checkout lines as time flows. To simulate the behavior and save the data of customers and cashier, I created Customer.[cc|h] and Casher.[cc|h] respectively.

   To get the required information as asked, I need to simulate the whole process. I focus on the actions in each minute. The process can be divided into 3 steps:
   1. Generate customers at random numbers: rand(0,5) at regular hour and rand(5,10) at peak hour.
   2. Arrange customers: 1) check for every casher if the customer can be served immediately or arranged to checkout line.

      2) if all the checkout lines are full (5 for each line), then try to assign to waiting line.

      3) if the number of customers in waiting line is 10 and there are still cashers not opened, then open a new casher and get 6 customers immediately.

      4) if all the cashers are opened, the unassigned customers will go to the waiting line.
   3. Advance one minute for each cashers, so the data will change and be recorded.

   There are also 2 steps to advance a minute in Casher:
   1. Determine if the last customer is finished serving.
   2. 1) If finished, dequeue the next customer from checkout line or waiting list and record the total serving time. If no one can be served, the casher will close.

      2) Move customer to checkout line to waiting list if it is not empty.
   3. Reduce the remain serving time.

   There is a watershed before advancing one minute and after to make sure all things work in order:
   1. Before advancing one minute, we assigned customers to different lines, so the arrival time is recorded as the Customer object created. We need to determine if in next minute a customer will left so that we can obtain all the information of that customer before it is deleted.
   2. After advancing the minute, all the states of data are changed and need to be recorded before next minute actually arriving.

After the simulation is built and run correctly, I recorded the required data at appropriate point.

- average number of customers arriving for checkout

  Once the number of arriving customer generated, the number will be added to sum which will be averaged after one hour.

- average/shortest/longest waiting time

  Waiting time = Arrival time – Start served time.

  The arrival time is recorded once the Customer object is created and the start served time is also recorded in Customer by Cahser object at the beginning of the serving.

- average/shortest/longest service time

  The service time is calculated and stored in Casher object. It will be added to sum once a customer is foreseen to leave.

- average number of open lines

  There are flags to indicate if a casher is open. So calculate them each minute and we can acquire the average number by divding by 60.

- maximum number of open lines

  The number of open lines will be compared for each minute.

- average time each casher will have more than 3 customers standing in line

  Set a flag to indicate if each casher has more than 3 customers standing in line. If the flag isn't changed by unsatisfied casher, then the counter will increment.

- average/smallest/largest number of customers in the waiting queue

  The number of customers in waiting queue will be record at the end of each minute.

For the entire simulation, the statistics can be acquired in similar methods.

Above all, the whole program to simulate the process is completely designed.

3. Implementation of the solution

In threadtest.cc

```
#include "kernel.h"
#include "main.h"
#include "thread.h"
#include "Simulator.h"

void
SimpleThread(int which)
{
   cout << "Start Simulating..\n";
   Simulator *sim = new Simulator();
   sim->Simulating();
}

void
ThreadTest()
{
   Thread *t = new Thread("forked thread");
   t->Fork((VoidFunctionPtr) SimpleThread, (void *) 1);
}
```

<u>In Simulator.h</u>

```
#pragma once
// Simulator.h
// Simulate the checkout process of the market from 14:00pm to 19:00pm.

#ifndef SIMULATOR_H
#define SIMULATOR_H

#include "kernel.h"
#include "main.h"
#include "thread.h"
#include "Casher.h"
#include "Customer.h"

class Simulator {
public:
  Simulator() { // constructor
  }
 int TimeInMinute(int hour, int minute);

 int OpenCasherCount(int arr[10]);

 int Min(int a, int b);

 int Max(int a, int b);

 void Simulating();
};

#endif // SIMULATOR_H
```

In Simulator.cc

```
#include "Simulator.h"

int Simulator::TimeInMinute(int hour, int minute){
  return 60 * (hour-14) + minute;
}

int Simulator::OpenCasherCount(int arr[10]) {
  int count = 0;
  for (int i = 0; i < 10; i++) {
    if (arr[i] == 1) {
      count++;
    }
  }
  return count;
}

int Simulator::Min(int a, int b) {
  if (a > b) {
    return b;
  }
  return a;
}

int Simulator::Max(int a, int b) {
  if (a > b) {
    return a;
  }
  return b;
}

void Simulator::Simulating() {
  Casher *casher_arr[10]; // array to store the addr of casher object
  int open_flag[10] = {0}; // flag to indicate if a checkout line is closed
  List<Customer*>  *waitingLine = new List<Customer*>(); // the single waiting line
  int waitingNum = 0;

  // initialize data, only 2 line open
  for(int i = 0; i < 10; i++){
    casher_arr[i] = new Casher(waitingLine, i);
  }
  open_flag[0] = 1;
  open_flag[1] = 1;

  // variables for entire simulation
  double total_avrWaitimgTimeSum = 0, total_avrServiceSum = 0;
  int total_minWaitingTime = 300, total_maxWaitingTime = 0, total_minServiceTime = 300,
total_maxServiceTime = 0;
  int total_maxWLNum = 0;

  // start simulating
  for (int hour = 14; hour < 19; hour++) { // simulating hours
    // variables for each hour
    double hour_totalArrivingNum = 0;
    int hour_cusServedNum = 0;
    int hour_maxWaitingTime = 0, hour_minWaitingTime = 10000, hour_totalWaitingTime = 0;
    int hour_maxServiceTime = 0, hour_minServiceTime = 300, hour_totalServiceTime = 0;
    int hour_maxOpenLineNum = 2, hour_totalOpenLineNum = 0;
    int hour_threeCusTime = 0, hour_threeCusTimeSum = 0;
    int hour_maxWLNum = 0, hour_minWLNum = 10000, hour_totalWLNum = 0;

    int offset = 0;
    cout << "[Time: " << hour << ":00 pm]";
    if (hour >= 16 && hour < 18) { // peak time
      offset = 5; //customer_Num = 5 + rand() % 6; // 5~10
      cout << "Peak Time\n" ;
    }
    else { // regular time
      offset = 0; //customer_Num = rand() % 6; // 0~5
      cout << "Regular Time\n";
    }

    for (int minute = 0; minute < 60; minute++) { //simulating minutes
      bool isFull = false;

      if (minute < 10){
        stringstream str;
        str << "-" << hour << ":0" << minute << "pm-";
        DEBUG('z', str.str());
      }else{
        stringstream str;
        str << "-" << hour << ":" << minute << "pm-";
        DEBUG('z', str.str());
      }

      // step #0, generate customer number
      int customer_Num = 0;
      customer_Num = offset + rand() % 6;
      stringstream str;
      str << "[" << customer_Num << " Customer(s) are generated.]";
      DEBUG('z', str.str());
      //customer_Num Customer(s) are generate
      hour_totalArrivingNum = hour_totalArrivingNum + customer_Num;

      // step #1, arrange customers
      // check for every casher
      while (isFull == false && customer_Num > 0){
        DEBUG('z', "--> Attempting to arrange to cashers");
        int fullNum = 0;
        for (int i = 0; i < 10; i++) {
          if (customer_Num == 0){
            break;
          }
          if (open_flag[i] == 1 && casher_arr[i]->getInLineNum() < 5) {
            int old = casher_arr[i]->getInLineNum();
            casher_arr[i]->addCustomerInLine(new Customer(TimeInMinute(hour,
minute)),TimeInMinute(hour, minute));
            stringstream cs_str;
            cs_str << "+ Casher " << i << " Checkout line from " << old << " -> " <<
casher_arr[i]->getInLineNum();
            DEBUG('z', cs_str.str());
            customer_Num--;
          }else if(open_flag[i] == 1 && casher_arr[i]->getInLineNum() == 5){
              fullNum++;
          }
          if(fullNum == OpenCasherCount(open_flag)){
             //casher opened
             isFull = true;
             break;
          }
        }
      }

      if (customer_Num > 0) { // there are customer not in queue
```

```
    // all checkout lines are full
    DEBUG('z', "--> Attempting to arrange to waiting line");
    if (waitingLine->NumInList() < 10) { // waiting queue not full
     int enqueNum = Min(customer_Num, 10 - waitingLine->NumInList());
     for (int i = 0; i < enqueNum; i++) {
       waitingLine->Append(new Customer(TimeInMinute(hour, minute)));
       customer_Num--;
     }
    }
   }

   while (customer_Num > 0) { // there are customer not in queue
    DEBUG('z',"--> Attempting to open new cashers");
    // all lines are full, add casher (must < 10)
    if (OpenCasherCount(open_flag) < 10) {
     for (int i = 0; i < 10; i++) {
       if (open_flag[i] == 0) {
        //cout << "New Casher " << i << " Opened!\n";
        //casher_arr[i] = new Casher(waitingLine);
        open_flag[i] = 1; // mark as opened
        int newNum = Min(5, customer_Num);
        for (int j = 0; j < 5; j++) {
          casher_arr[i]->addCustomerInLine(waitingLine->RemoveFront(), TimeInMinute(hour,
minute)); // arrange waiting list to new casher
          casher_arr[i]->getCus()->setStartTime(TimeInMinute(hour, minute));
          if (j < newNum){ // arrange new customers to waiting list
            waitingLine->Append(new Customer(TimeInMinute(hour, minute)));
            customer_Num --;
          }
        }
        break;
       }
     }
    }
    else {
     for (int i = 0; i < customer_Num; i++) {
       waitingLine->Append(new Customer(TimeInMinute(hour, minute)));
     }
     //All Cashers are full
     break;
    }
   }

   stringstream w_str;
   w_str << waitingLine->NumInList() << " in waitng line!";
   DEBUG('z',w_str.str());
   DEBUG('z',"---------------Advancing One Minute------------------");
   bool allThreeFlag = true;
   // step #2, advance 1 minute
   for (int i = 0; i < 10; i++) {
    if (open_flag[i] == 1) { // only process the open line

     if (casher_arr[i]->getRemainTime() == 0 && casher_arr[i]->getCus() != NULL){ // last
customer left, record data.
       hour_cusServedNum ++;
       int newRecord = casher_arr[i]->getCus()->getStartTime() -
casher_arr[i]->getCus()->getArrivalTime();
       hour_minWaitingTime = Min(hour_minWaitingTime, newRecord);
       hour_maxWaitingTime = Max(hour_maxWaitingTime, newRecord);
       hour_totalWaitingTime = hour_totalWaitingTime + newRecord;

       newRecord = casher_arr[i]->getTotalTime();
       hour_minServiceTime = Min(hour_minServiceTime, newRecord);
       hour_maxServiceTime = Max(hour_maxServiceTime, newRecord);
       hour_totalServiceTime = hour_totalServiceTime + newRecord;

       hour_maxOpenLineNum = Max(hour_maxOpenLineNum, OpenCasherCount(open_flag));
     }

     if (casher_arr[i]->NextMove(TimeInMinute(hour, minute)) == -1) { // advance one minute for
the casher, check whether the casher need to be closed
       if (OpenCasherCount(open_flag) > 2) { // keep minimum checkout line num
        open_flag[i] = 0;
       }
     }
     if (allThreeFlag == false || casher_arr[i]->getInLineNum() <= 3){
       allThreeFlag = false;
     }
     stringstream c_str;
     c_str << "Casher @" << i << " has #" << casher_arr[i]->getInLineNum() << "# in wating line!";
     DEBUG('z', c_str.str());
    }
   }
    stringstream wt_str;
    wt_str << waitingLine->NumInList() << " in waitng line!\n";
    DEBUG('z',wt_str.str());
    int newRecord = waitingLine->NumInList();
    hour_minWLNum = Min(hour_minWLNum, newRecord);
    hour_maxWLNum = Max(hour_maxWLNum, newRecord);
    hour_totalWLNum = hour_totalWLNum + newRecord;

    if (allThreeFlag == true){
      hour_threeCusTime ++;
    }

   hour_totalOpenLineNum = hour_totalOpenLineNum + OpenCasherCount(open_flag);
   stringstream d_str;
   d_str << "Minute summary:\n" << "- Average/shortest/longest waiting time: - / " <<
hour_minWaitingTime << " / " << hour_maxWaitingTime << "\n" << "- Average/shortest/longest
service time: - / " << hour_minServiceTime << " / " << hour_maxServiceTime << "\n" << "-
Maximum number of open lines: " << hour_maxOpenLineNum << "\n - Average/smallest/largest
number of customers in the waiting queue: - / " << hour_minWLNum << " / " << hour_maxWLNum
<< "\n";
   DEBUG('z', d_str.str());

   } // minute

   total_avrWaitimgTimeSum = total_avrWaitimgTimeSum + hour_totalWaitingTime /
(double)hour_cusServedNum;
   total_avrServiceSum = total_avrServiceSum + hour_totalServiceTime /
(double)hour_cusServedNum;
   total_minWaitingTime = Min(total_minWaitingTime, hour_minWaitingTime);
   total_maxWaitingTime = Max(total_maxWaitingTime, hour_maxWaitingTime);
   total_minServiceTime = Min(total_minServiceTime, hour_minServiceTime);
   total_maxServiceTime = Max(total_maxServiceTime, hour_maxServiceTime);
   total_maxWLNum = Max(total_maxWLNum, hour_maxWLNum);

   cout << "Hour summary:\n";
   cout << "- Average number of customers arriving for checkout: " <<
hour_totalArrivingNum/60.00 << "\n";
   cout << "- Average/shortest/longest waiting time: " << hour_totalWaitingTime /
(double)hour_cusServedNum << " / " << hour_minWaitingTime << " / " << hour_maxWaitingTime
<< "\n";
   cout << "- Average/shortest/longest service time: " << hour_totalServiceTime /
(double)hour_cusServedNum << " / " << hour_minServiceTime << " / " << hour_maxServiceTime <<
"\n";
   cout << "- Average number of open lines: " << hour_totalOpenLineNum / 60.00 << " \n";
   cout << "- Maximum number of open lines: " << hour_maxOpenLineNum << " \n";
```

```
    cout << "- Average time each casher will have more than 3 customers standing in line: " <<
hour_threeCusTime << " \n";
    cout << "- Average/smallest/largest number of customers in the waiting queue: " <<
hour_totalWLNum / 60.00 << " / " << hour_minWLNum << " / " << hour_maxWLNum << "\n\n";
  } // hour
  cout << "\n-------------------------------------------------------------------------------\n\n";
  cout << "Total summary:\n";
  cout << "- Average/shortest/longest waiting time: " << total_avrWaitimgTimeSum / 5.00 << " / "
<< total_minWaitingTime << " / " << total_maxWaitingTime << "\n";
  cout << "- Average/shortest/longest service time: " << total_avrServiceSum / 5.00 << " / " <<
total_minServiceTime << " / " << total_maxServiceTime << "\n";
  cout << "- Maximum number of customers in the waiting queue at any time: " << total_maxWLNum
<< " \n";
}In Customer.h
#pragma once
// Customer.h
// Simulate the behavour of the customer and generate relating data.

#ifndef CUSTOMER_H
#define CUSTOMER_H

#include <stdlib.h>

class Customer {
public:
  Customer(int arrTime) { // constructor
    // generate item num
    _itemNum = 5 + rand() % 36;
    _arrivalTime = arrTime;
  }
  void setStartTime(int startTime);
  int getStartTime();

  int getItemNum();
  int getArrivalTime();
private:
  int _itemNum;
  int _arrivalTime; // record in minutes
  int _startServiceTime;
};

#endif // CUSTOMER_H
```

<u>In Customer.cc</u>

```
#include "Customer.h"

int Customer::getItemNum() {
  return _itemNum;
}

int Customer::getArrivalTime(){
  return _arrivalTime;
}


void Customer::setStartTime(int startTime){
  _startServiceTime = startTime;
}

int Customer::getStartTime(){
  return _startServiceTime;
}
```

3. Testing

1. Method to run my tests.
   1) put threadtest.cc, Simulator.h, Simulator.cc, Casher.h, Casher.cc, Customer.h, Customer.cc into ../code/threads. Put makefile into the directory of build.
   2) open to working folder, using command line to compile:

      ```
      make depend
      make
      ```
   3) Run:

      [result mode] **nachos -K** => only required result

      [detail mode] **nachos -d -z -K** => detail information for each minute

2. When verifying the simulation process, I printed the status of casher and each line in every minute.

The following information of 7 minutes can prove that the way to arrange customers is in correct order (printed from detail mode):

```
[Time: 14:00 pm]Regular Time
-14:00pm-
[1 Customer(s) are generated.]
--> Attempting to arrange to cashers
[Casher 0]New Customer! [Waiting Time: 0][Total Service Time: 5]
+ Casher 0 Checkout line from 0 -> 0
0 in waitng line!
---------------Advancing One Minute------------------
[Casher 0]Remain Service Time: {4}
Casher @0 has #0# in wating line!
Casher @1 has #0# in wating line!
0 in waitng line!

Minute summary:
- Average/shortest/longest waiting time: - / 10000 / 0
- Average/shortest/longest service time: - / 300 / 0
- Maximum number of open lines: 2
 - Average/smallest/largest number of customers in the waiting queue: - / 0 / 0

-14:01pm-
[3 Customer(s) are generated.]
--> Attempting to arrange to cashers
+ Casher 0 Checkout line from 0 -> 1
[Casher 1]New Customer! [Waiting Time: 0][Total Service Time: 3]
+ Casher 1 Checkout line from 0 -> 0
--> Attempting to arrange to cashers
```

```
+ Casher 0 Checkout line from 1 -> 2
0 in waitng line!
---------------Advancing One Minute------------------
[Casher 0]Remain Service Time: {3}
Casher @0 has #2# in wating line!
[Casher 1]Remain Service Time: {2}
Casher @1 has #0# in wating line!
0 in waitng line!

Minute summary:
- Average/shortest/longest waiting time: - / 10000 / 0
- Average/shortest/longest service time: - / 300 / 0
- Maximum number of open lines: 2
 - Average/smallest/largest number of customers in the waiting queue: - / 0 / 0

-14:02pm-
[4 Customer(s) are generated.]
--> Attempting to arrange to cashers
+ Casher 0 Checkout line from 2 -> 3
+ Casher 1 Checkout line from 0 -> 1
--> Attempting to arrange to cashers
+ Casher 0 Checkout line from 3 -> 4
+ Casher 1 Checkout line from 1 -> 2
0 in waitng line!
---------------Advancing One Minute------------------
[Casher 0]Remain Service Time: {2}
Casher @0 has #4# in wating line!
[Casher 1]Remain Service Time: {1}
Casher @1 has #2# in wating line!
0 in waitng line!

Minute summary:
- Average/shortest/longest waiting time: - / 10000 / 0
- Average/shortest/longest service time: - / 300 / 0
- Maximum number of open lines: 2
 - Average/smallest/largest number of customers in the waiting queue: - / 0 / 0

-14:03pm-
[1 Customer(s) are generated.]
--> Attempting to arrange to cashers
+ Casher 0 Checkout line from 4 -> 5
```

```
0 in waitng line!
---------------Advancing One Minute------------------
[Casher 0]Remain Service Time: {1}
Casher @0 has #5# in wating line!
[Casher 1]Remain Service Time: {0}
Casher @1 has #2# in wating line!
0 in waitng line!

Minute summary:
- Average/shortest/longest waiting time: - / 10000 / 0
- Average/shortest/longest service time: - / 300 / 0
- Maximum number of open lines: 2
 - Average/smallest/largest number of customers in the waiting queue: - / 0 / 0

-14:04pm-
[1 Customer(s) are generated.]
--> Attempting to arrange to cashers
+ Casher 1 Checkout line from 2 -> 3
0 in waitng line!
---------------Advancing One Minute------------------
[Casher 0]Remain Service Time: {0}
Casher @0 has #5# in wating line!
[Casher 1]New Customer! [Waiting Time: 2][Total Service Time: 5]
[Casher 1]Remain Service Time: {4}
Casher @1 has #2# in wating line!
0 in waitng line!

Minute summary:
- Average/shortest/longest waiting time: - / 0 / 0
- Average/shortest/longest service time: - / 3 / 3
- Maximum number of open lines: 2
 - Average/smallest/largest number of customers in the waiting queue: - / 0 / 0

-14:05pm-
[4 Customer(s) are generated.]
--> Attempting to arrange to cashers
+ Casher 1 Checkout line from 2 -> 3
--> Attempting to arrange to cashers
+ Casher 1 Checkout line from 3 -> 4
--> Attempting to arrange to cashers
+ Casher 1 Checkout line from 4 -> 5
```

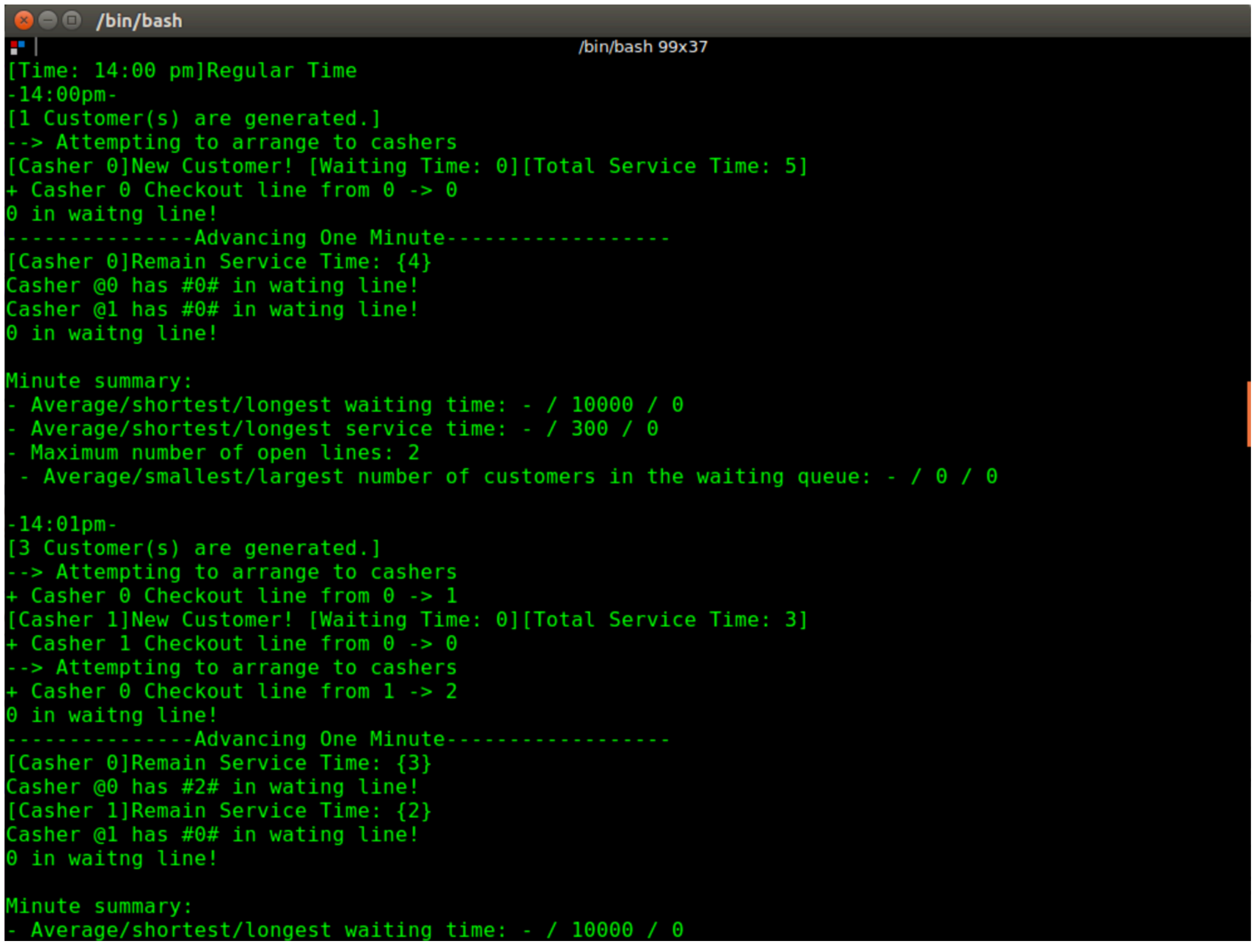
```
--> Attempting to arrange to cashers
--> Attempting to arrange to waiting line
1 in waitng line!
---------------Advancing One Minute------------------
[Casher 0]New Customer! [Waiting Time: 4][Total Service Time: 3]
[Casher 0] 1 Customer added to checkout list by casher!
[Casher 0]Remain Service Time: {2}
Casher @0 has #5# in wating line!
[Casher 1]Remain Service Time: {3}
Casher @1 has #5# in wating line!
0 in waitng line!

Minute summary:
- Average/shortest/longest waiting time: - / 0 / 0
- Average/shortest/longest service time: - / 3 / 5
- Maximum number of open lines: 2
 - Average/smallest/largest number of customers in the waiting queue: - / 0 / 0

-14:06pm-
[5 Customer(s) are generated.]
--> Attempting to arrange to cashers
--> Attempting to arrange to waiting line
5 in waitng line!
---------------Advancing One Minute------------------
[Casher 0]Remain Service Time: {1}
Casher @0 has #5# in wating line!
[Casher 1]Remain Service Time: {2}
Casher @1 has #5# in wating line!
5 in waitng line!

Minute summary:
- Average/shortest/longest waiting time: - / 0 / 0
- Average/shortest/longest service time: - / 3 / 5
- Maximum number of open lines: 2
 - Average/smallest/largest number of customers in the waiting queue: - / 0 / 5
```

3. The required result (result mode):

```
[Time: 14:00 pm]Regular Time
Hour summary:
- Average number of customers arriving for checkout: 2.55
- Average/shortest/longest waiting time: 15.4468 / 0 / 26
- Average/shortest/longest service time: 3.94681 / 3 / 5
- Average number of open lines: 6.46667
- Maximum number of open lines: 10
- Average time each casher will have more than 3 customers standing in line: 55
- Average/smallest/largest number of customers in the waiting queue: 4.81667 / 0 / 10

[Time: 15:00 pm]Regular Time
Hour summary:
- Average number of customers arriving for checkout: 2.25
- Average/shortest/longest waiting time: 20.4333 / 17 / 26
- Average/shortest/longest service time: 3.88 / 3 / 5
- Average number of open lines: 10
- Maximum number of open lines: 10
- Average time each casher will have more than 3 customers standing in line: 56
- Average/smallest/largest number of customers in the waiting queue: 2.21667 / 0 / 7

[Time: 16:00 pm]Peak Time
Hour summary:
- Average number of customers arriving for checkout: 7.45
- Average/shortest/longest waiting time: 24.329 / 9 / 41
- Average/shortest/longest service time: 3.91613 / 3 / 5
- Average number of open lines: 10
- Maximum number of open lines: 10
- Average time each casher will have more than 3 customers standing in line: 56
- Average/smallest/largest number of customers in the waiting queue: 126.9 / 0 / 276

[Time: 17:00 pm]Peak Time
Hour summary:
- Average number of customers arriving for checkout: 7.7
- Average/shortest/longest waiting time: 60.098 / 39 / 81
- Average/shortest/longest service time: 3.92157 / 3 / 5
- Average number of open lines: 10
- Maximum number of open lines: 10
- Average time each casher will have more than 3 customers standing in line: 60
- Average/smallest/largest number of customers in the waiting queue: 430.833 / 281 / 585
```

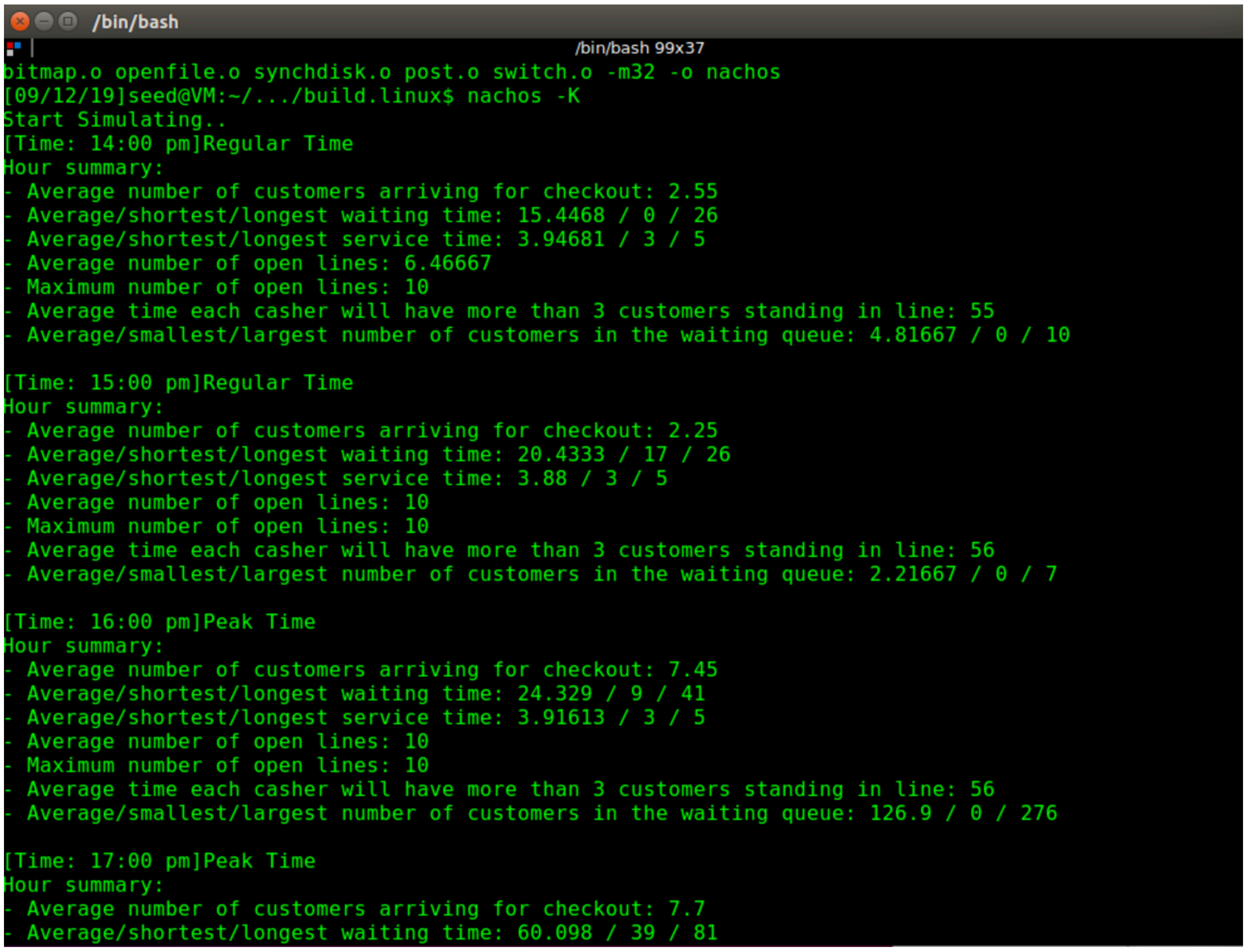
```
[Time: 18:00 pm]Regular Time
Hour summary:
- Average number of customers arriving for checkout: 2.35
- Average/shortest/longest waiting time: 99.9932 / 80 / 121
- Average/shortest/longest service time: 4.10811 / 3 / 5
- Average number of open lines: 10
- Maximum number of open lines: 10
- Average time each casher will have more than 3 customers standing in line: 60
- Average/smallest/largest number of customers in the waiting queue: 585.267 / 578 / 589


-------------------------------------------------------------------------------

Total summary:
- Average/shortest/longest waiting time: 44.0601 / 0 / 121
- Average/shortest/longest service time: 3.95452 / 3 / 5
- Maximum number of customers in the waiting queue at any time: 589
```

4. Output Snapshots



```
/bin/bash
/bin/bash 99x37
[Time: 14:00 pm]Regular Time
-14:00pm-
[1 Customer(s) are generated.]
--> Attempting to arrange to cashers
[Casher 0]New Customer! [Waiting Time: 0][Total Service Time: 5]
+ Casher 0 Checkout line from 0 -> 0
0 in waitng line!
---------------Advancing One Minute------------------
[Casher 0]Remain Service Time: {4}
Casher @0 has #0# in wating line!
Casher @1 has #0# in wating line!
0 in waitng line!

Minute summary:
- Average/shortest/longest waiting time: - / 10000 / 0
- Average/shortest/longest service time: - / 300 / 0
- Maximum number of open lines: 2
 - Average/smallest/largest number of customers in the waiting queue: - / 0 / 0

-14:01pm-
[3 Customer(s) are generated.]
--> Attempting to arrange to cashers
+ Casher 0 Checkout line from 0 -> 1
[Casher 1]New Customer! [Waiting Time: 0][Total Service Time: 3]
+ Casher 1 Checkout line from 0 -> 0
--> Attempting to arrange to cashers
+ Casher 0 Checkout line from 1 -> 2
0 in waitng line!
---------------Advancing One Minute------------------
[Casher 0]Remain Service Time: {3}
Casher @0 has #2# in wating line!
[Casher 1]Remain Service Time: {2}
Casher @1 has #0# in wating line!
0 in waitng line!

Minute summary:
- Average/shortest/longest waiting time: - / 10000 / 0
```

Fig 1 Detail Mode



/bin/bash

/bin/bash 99x37

```
bitmap.o openfile.o synchdisk.o post.o switch.o -m32 -o nachos
[09/12/19]seed@VM:~/.../build.linux$ nachos -K
Start Simulating..
[Time: 14:00 pm]Regular Time
Hour summary:
- Average number of customers arriving for checkout: 2.55
- Average/shortest/longest waiting time: 15.4468 / 0 / 26
- Average/shortest/longest service time: 3.94681 / 3 / 5
- Average number of open lines: 6.46667
- Maximum number of open lines: 10
- Average time each casher will have more than 3 customers standing in line: 55
- Average/smallest/largest number of customers in the waiting queue: 4.81667 / 0 / 10

[Time: 15:00 pm]Regular Time
Hour summary:
- Average number of customers arriving for checkout: 2.25
- Average/shortest/longest waiting time: 20.4333 / 17 / 26
- Average/shortest/longest service time: 3.88 / 3 / 5
- Average number of open lines: 10
- Maximum number of open lines: 10
- Average time each casher will have more than 3 customers standing in line: 56
- Average/smallest/largest number of customers in the waiting queue: 2.21667 / 0 / 7

[Time: 16:00 pm]Peak Time
Hour summary:
- Average number of customers arriving for checkout: 7.45
- Average/shortest/longest waiting time: 24.329 / 9 / 41
- Average/shortest/longest service time: 3.91613 / 3 / 5
- Average number of open lines: 10
- Maximum number of open lines: 10
- Average time each casher will have more than 3 customers standing in line: 56
- Average/smallest/largest number of customers in the waiting queue: 126.9 / 0 / 276

[Time: 17:00 pm]Peak Time
Hour summary:
- Average number of customers arriving for checkout: 7.7
- Average/shortest/longest waiting time: 60.098 / 39 / 81
```

Fig 2 Result Mode