Міністерство освіти і науки України

Національний університет «Запорізька Політехніка»

Кафедра програмних засобів

ЗВІТ

з лабораторної роботи №2

з дисципліни «Моделювання та Аналіз Програмного Забезпечення» на тему:

«Моделювання систем масового обслуговування з одним обслуговуючим приладом та чергою»

Виконав:

Студент групи КНТ-122 О. А. Онищенко

Прийняли:

Викладач: Ж. К. Камінська

2024

МОДЕЛЮВАННЯ СИСТЕМ МАСОВОГО ОБСЛУГОВУВАННЯ З ОДНИМ ОБСЛУГОВУЮЧИМ ПРИЛАДОМ ТА ЧЕРГОЮ

Мета роботи

Вивчити методи моделювання різних дисциплін обслуговування в СМО з одним приладом та чергою, аналізувати вихідні дані моделювання з метою вибору оптимального варіанту реалізації системи, що моделюється.

Індивідуальне завдання

У механічний цех, відразу ж з початком робочого дня, кожні 3хв надходять червноі деталі з масою 2кг, кожні 5хв червоні деталі з масою 2,4кг і кожні 20хв зелені з масою 3кг та температурою 85С і двома отворами. Червоні деталі надходять для обробки на свердлильному верстаті. У кожній з них свердлять по одному отвору. Час обслуговування: червоні – 15хв. Зелені поступають на токарний верстат і обробляються 20хв. Час встановлення на верстат та зняття з нього становить 4хв для кожної деталі. Промоделювати роботу цеху протягом 8 годин робочого дня. Визначити чергу перед свердлильним верстатом.

Опис термінів

Прилад - динамічний об'єкт для моделювання апарату, що обслуговує транзакти

Черга - динамічний об'єкт для просування транзактів на ділянках моделі та статистики

Результати виконання

Код програми

Загальне завдання

```
#include "../simc/simc.h"
#include <iostream>
using namespace std;

int TOTAL_HOURS = 28800;
int FIRST_INTERVAL = 510;
int FIRST_DURATION = 300;
int SECOND_INTERVAL = 420;
int SECOND_DURATION = 100;

pair<double, double> model(bool givePriorityToYellow = false)
{
  pqueue partsQueue;
  pfacility warehousePerson;
  int totalPeopleInQueue = 0;
  int totalIterations = 0;

  initlist(TOTAL_HOURS);
  initcreate(1, 0);
  initcreate(8, 0);
  newqueue(partsQueue, "\"Orders Order\"");
  newfac(warehousePerson, "\"Warehouse\"");
  while (systime < TOTAL_HOURS)
  {
    plan();
    switch (sysevent)
    {
    case 1:
      create(FIRST_INTERVAL);
      break;
    case 2:
      inqueue(partsQueue);
      trans->prty = 1;
      totalPeopleInQueue += 1;
      break;
    case 3:
      seize(warehousePerson);
      break;
    case 4:
      outqueue(partsQueue);
      totalPeopleInQueue -= 1;
      break;
```

```
    case 5:
      delayt(FIRST_DURATION);
      break;
    case 6:
      outfac(warehousePerson);
      break;
    case 7:
      destroy();
      break;

    case 8:
      create(SECOND_INTERVAL);
      break;
    case 9:
      inqueue(partsQueue);
      if (givePriorityToYellow == true)
      {
        trans->prty = 2;
      }
      else
      {
        trans->prty = 1;
      }
      totalPeopleInQueue += 1;
      break;
    case 10:
      seize(warehousePerson);
      break;
    case 11:
      outqueue(partsQueue);
      totalPeopleInQueue -= 1;
      break;
    case 12:
      delayt(SECOND_DURATION);
      break;
    case 13:
      outfac(warehousePerson);
      break;
    case 14:
      destroy();
      break;
    }
    totalIterations += 1;
  }
  printall();
  double averagePeopleInQueue = (double)totalPeopleInQueue /
totalIterations;
```

```
  double totalMoneyLost = totalPeopleInQueue * 0.25 * FIRST_DURATION +
totalPeopleInQueue * 0.25 * SECOND_DURATION;
  return make_pair(averagePeopleInQueue, totalMoneyLost);
}
int main()
{
  pair<double, double> resOne = model(false);
  pair<double, double> resTwo = model(true);

  cout << endl
       << "Average people in queue without prioritization: " <<
resOne.first << endl;
  cout << "Average people in queue with prioritization: " << resTwo.first
<< endl;

  cout << "Total money lost without prioritization: " << resOne.second <<
endl;
  cout << "Total money lost with prioritization: " << resTwo.second <<
endl;

  return 0;
}
```

Індивідуальне завдання

```
#include "../simc/simc.h"
#include <iostream>
using namespace std;

auto STARTING_TIME = 0;
auto TIME_TO_TAKE_PARTS = 4;
auto TIME_TO_WORK = 480;

auto RED_MASS = 2;
auto RED_INTERVAL = 3;
auto RED_HOLES = 0;

auto HEFTY_RED_MASS = 2.4;
auto HEFTY_RED_INTERVAL = 5;
auto HEFTY_RED_HOLES = 0;

auto GREEN_MASS = 3;
auto GREEN_INTERVAL = 20;
auto GREEN_TEMPERATURE = 85;
auto GREEN_HOLES = 2;
```

```
auto DRILLING_MACHINE_DURATION = 15;
auto LATHE_DURATION = 20;
auto DRILLING_MACHINE_NAME = "\"Drilling Machine\"";
auto LATHE_NAME = "\"Lathe\"";

auto TOTAL_WORKING_TIME = 0;
auto TOTAL_DRILLING_MACHINE_TIME = 0;
auto TOTAL_LATHE_TIME = 0;
auto TOTAL_HOLES = 0;

void model()
{
  pstorage drillingMachine;
  pstorage lathe;

  initlist(TIME_TO_WORK);
  // red drill
  initcreate(1, STARTING_TIME);
  // hefty red drill
  initcreate(4, STARTING_TIME);
  // green lathe
  initcreate(7, STARTING_TIME);

  newstorage(drillingMachine, DRILLING_MACHINE_NAME, HEFTY_RED_INTERVAL);
  newstorage(lathe, LATHE_NAME, GREEN_INTERVAL);

  const int drillDelay = TIME_TO_TAKE_PARTS + DRILLING_MACHINE_DURATION;
  const int latheDelay = TIME_TO_TAKE_PARTS + LATHE_DURATION;
  cout << "Drilling machine queue: " << endl;

  while (systime < TIME_TO_WORK)
  {
    plan();
    switch (sysevent)
    {
    case 1:
      enter(drillingMachine, 1);
      TOTAL_HOLES += 1;
      cout << "Transact " << trans->nom << " in queue" << endl;
      break;
    case 2:
      delayt(drillDelay);
      TOTAL_WORKING_TIME += drillDelay;
      TOTAL_DRILLING_MACHINE_TIME += drillDelay;
      break;
    case 3:
      leave(drillingMachine, 1);
      cout << "Transact " << trans->nom << " out of queue" << endl;
```

```
      break;

    case 4:
      enter(drillingMachine, 1);
      TOTAL_HOLES += 1;
      cout << "Transact " << trans->nom << " in queue" << endl;
      break;
    case 5:
      delayt(drillDelay);
      TOTAL_WORKING_TIME += drillDelay;
      TOTAL_DRILLING_MACHINE_TIME += drillDelay;
      break;
    case 6:
      leave(drillingMachine, 1);
      cout << "Transact " << trans->nom << " out of queue" << endl;
      break;

    case 7:
      enter(lathe, 1);
      cout << "Transact " << trans->nom << " in queue" << endl;
      break;
    case 8:
      delayt(latheDelay);
      TOTAL_WORKING_TIME += latheDelay;
      TOTAL_LATHE_TIME += latheDelay;
      break;
    case 9:
      leave(lathe, 1);
      cout << "Transact " << trans->nom << " out of queue" << endl;
      break;

    case 10:
      next(1);
      cout << endl;
      break;
    }
  }
  printall();
  clear();
  destrs(drillingMachine);
  cout << endl
       << "Modeling finished, praise Jesus Christ our Holy Lord Almighty
Living GOD Most High" << endl
       << "  - Total time the machines worked: " << TOTAL_WORKING_TIME <<
" minutes" << endl
       << "  - Total time the drill worked: " <<
TOTAL_DRILLING_MACHINE_TIME << " minutes" << endl
```

```
        << "  - Total time the lathe worked: " << TOTAL_LATHE_TIME << "
minutes" << endl
        << "  - Total number of holes: " << TOTAL_HOLES << endl;
}
int main()
{
  model();
  return 0;
}
// g++ ind.cpp ../simc/simc.cpp -o ind
```

Виконання програми

SIM-CI++ v1.2	НУ"ЗП"	2024
Ж. К. Камінська С. М. Сердюк		

Загальні параметри середовища:

Поточний час	28860.000
Поточна подія	6
Поточний транзакт	4
Усього подій	929.000
Час моделювання	0.00 сек.
Середній час виконання події	0.00000 сек/подія

ПОДІЯ	1	2	3	4	5	6	7	8	9	10
УСЬОГО	57	57	69	57	57	57	56	69	69	109

ПОДІЯ	11	12	13	14	15	16	17	18	19	20
УСЬОГО	68	68	68	68						

Черги

Черга	Кількість входжень З нульовим часом очікування	Макс. довжина Поточна довжина	Середній час очікування Без урахування нульових входжень	Середня довжина	% входжень у порожню чергу
"Orders Order"	126 73	1 1	56.640 133.585	0.248	57.937

Прилади

Прилад	Кількість входжень	Середній час обробки	Завантаження	Кількість захоплень	Стан
"Warehouse"	125	191.200	0.828	0	FREE

Рисунок 1.1 – Загальне завдання – браузер

```
  Enter a name for the Report HTML file: gen
  Enter a name for the Report HTML file: gen

 Average people in queue without prioritization: 0.00107643
 Average people in queue with prioritization: 0.00107643
 Total money lost without prioritization: 100
 Total money lost with prioritization: 100
```

Рисунок 1.2 – Загальне завдання – консолька ✝

CIM-CI++ v1.2	НУ"ЗП"	2024
Ж. К. Камінська С. М. Сердюк		

Загальні параметри середовища:

Поточний час	496.000
Поточна подія	9
Поточний транзакт	1
Усього подій	237.000
Час моделювання	0.00 сек.
Середній час виконання події	0.00000 сек/подія

ПОДІЯ	1	2	3	4	5	6	7	8	9	10
УСЬОГО	24	24	23	24	24	23	24	24	24	23

Накопичувачі

Накопичувач	Ємність	Завантаженість	Середній час перебування	Вміст			Кількість входжень
				Поточн.	Макс.	Середн.	
"Drilling Machine"	5	0.366	18.596	2	3	1.83	48
"Lathe"	20	0.058	24.000	0	2	1.16	24

Рисунок 1.3 – Індивідуальне завдання – браузер

```
Transact 2 in queue
Transact 3 out of queue
Transact 3 in queue
Transact 1 out of queue

Transact 1 in queue
Transact 2 out of queue
Transact 2 in queue
Transact 3 out of queue
Transact 3 in queue
Transact 1 out of queue
Transact 1 in queue
Transact 2 out of queue
Transact 2 in queue
Transact 3 out of queue

Transact 3 in queue
Transact 1 out of queue
Transact 1 in queue
Transact 2 out of queue

Transact 2 in queue
Transact 3 out of queue
Transact 3 in queue
Transact 1 out of queue

Modeling finished, praise Jesus Christ our Holy Lord Almighty Living GOD Most High
  - Total time the machines worked: 1488 minutes
  - Total time the drill worked: 912 minutes
  - Total time the lathe worked: 576 minutes
  - Total number of holes: 48
```

Рисунок 1.4 – Індивідуальне завдання – консолька

Висновки

Таким чином, ми вивчили методи моделювання різних дисциплін обслуговування в СМО з одним приладом та чергою; а також проаналізували вихідні дані моделювання з метою вибору оптимального варіанту реалізації системи, що моделюється.

Контрольні питання

Основні дисципліни обслуговування в СМО

Основними дисциплінами обслуговування в Системах Масового Обслуговування є First in First out (FIFO) та Last in First out (LIFO). Також існують інші нестандартні дисципліни.

Множинні типи даних "ПРИЛАД" та "ЧЕРГА" СІМ

Прилад -динамічний об'єкт для моделювання апарату, що обслуговує транзакти. Транзакт може очікувати входу до приладу, займати чи звільняти його.

Черга -динамічний об'єкт для просування транзактів на ділянках моделі та статистики.

Процедури створення, знищення, реєстрації черги

Для створення черги можна використати void newqueue(посилання на чергу, ймення черги); для реєстрації черги або void inqueue(pqueue), або void outqueue(pqueue); для знищення черги можна використати void destrq(pqueue).