Московский Авиационный Институт
(Национальный Исследовательский Университет)
Факультет информационных технологий и прикладной математики
Кафедра вычислительной математики и программирования

Лабораторная работа №3 по курсу

«Операционные системы»

Тема работы

Управление потоками в ОС

Студент: Велесов Даниил Игоревич
Группа: М8О-208Б-20
Вариант:
Преподаватель: Миронов Евгений Сергеевич
Оценка: _____
Дата: _____
Подпись: _____

Москва, 2021

Содержание

Репозиторий

https://github.com/Kalambur4k/OS/tree/main/lab3

Постановка задачи

Составить программу на языке Си, обрабатывающую данные в многопоточном режиме. При обработки использовать стандартные средства создания потоков операционной системы (Windows/Unix). Ограничение потоков должно быть задано ключом запуска вашей программы.

Вариант 14

> Есть колода из 52 карт, рассчитать экспериментально (метод Монте-Карло) вероятность того, что сверху лежат две одинаковых карты(по значимости). Количество раундов подается с ключом

Общие сведения о программе

Программа состоит из одного файла card52.c в котором и происходит вся работа программы (расчёт вероятности выпадения двух одинаковых карт).

Общий метод и алгоритм решения

Для начала кодируем колоду от 0 до 51; Масти идут в следующем порядке: пики, кресты, бубны, червы. В каждой масти 13 карт, следовательно значение карты = номер карты % 13. Основная вычислительная функция void *calc_probability(void *arg), будет выполнять перетасовку карт и смотреть последние два числа. Если они будут “одной масти” то такой раунд мы посчитаем успешным, в противном случае раунд в счет успешных не пойдет. В main организована основная работа: организация многопоточной работы calc_probability где количество раундов и потоков вводится пользователем. После выполнения всех вычислений идет подсчет количества успешных раундов и их соотношение с проваленными (таким образом и получаем вероятность)

Исходный код

card52.c

```
#include <pthread.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>

#define NCARDS 52
#define SUIT_SIZE 13

void *calc_probability(void *arg) {
   long rounds = *(long*)arg;
   long *success;
   success = (long*) malloc(sizeof(long));
   *success = 0;

   int cards[NCARDS];
   int i;
   for ( i = 0; i < NCARDS - 1; i++ ) {
      cards[i] = i;
   }
   unsigned int seed = time(NULL) + (unsigned int) pthread_self();
   long j;
   for ( j = 0; j < rounds; j++ ) {
      int card1 = rand_r(&seed) % NCARDS;
      cards[card1] = NCARDS - 1;
      int card2 = rand_r(&seed) % ( NCARDS - 1 );
      if ( cards[card2] % SUIT_SIZE == card1 % SUIT_SIZE ) {
         (*success)++;
      }
      cards[card1] = card1;
   }

   pthread_exit(success);
}

int main(int argc, char* argv[] ) {

   if ( argc < 2 ) {
      printf( "Usage: %s n_threads n_rounds_per_thread\n", argv[0] );
      return 0;
   }

   int n_threads = atoi(argv[1]);
   long n_rounds = atol(argv[2]);

   printf( "%d threads and %ld rounds, PID = %d\n", n_threads, n_rounds, (int)getpid());

   pthread_t *threads = (pthread_t*) malloc(n_threads*sizeof(pthread_t));
   int *thread_ok = (int*) malloc(n_threads*sizeof(int));

   int i;
   for ( i = 0; i < n_threads; i++ ) {
      thread_ok[i] = pthread_create( &(threads[i]), NULL, calc_probability, (void*) &n_rounds );
```

```
    }

    long long total_rounds = 0, total_success = 0;
    for ( i = 0; i < n_threads; i++ ) {
        if ( !thread_ok[i] ) {
            void *res;
            pthread_join( threads[i], &res );
            total_rounds += n_rounds;
            total_success += *(long*)res;
            free( res );
        } else {
            printf( "thread %d error\n", i );
        }
    }

    long double result = 1.0L*total_success/total_rounds;

    printf( "%lld total rounds %lld total success\ncalculated result = %.15Lf\n", total_rounds,
total_success, result );
    printf( "ideal result      = %.15Lf\n", 3.0L/51 );

    free(threads);
    free(thread_ok);
}
```

Демонстрация работы программы

$ ./card52 10 20000000

10 threads and 20000000 rounds, PID = 8306

200000000 total rounds 11762583 total success

calculated result = 0.058812915000000

ideal result   = 0.058823529411765

Выводы

Благодаря этой лабораторной работе я научился работать и писать программы, работающие в многопоточном режиме, и применять эти знания на практике.