دانشگاه صنعتی امیرکبیر

(پلی تکنیک تهران)

دانشکده مهندسی برق

آزمایشگاه سیستم عامل

آزمایش ۵

فرایند ها و نخ ها

نگارش

علی بابالو

پویا شریفی

استاد راهنما

مهندس کیخا

خرداد ماه 1402

بخش اول:

طبق دستور کار آرایه hist با طول25 گرفته شد و به ازای iteration های مختلف مقدار آرایه ای با اندیس counter تغییر پیدا کرد. مقدار counter در هر مرحله به این صورت به دست می آید که 12 مرحله عددی رندم بین 0 تا 100 ایجاد می کنیم و اگر این عدد تصادفی بیش از 48 بود مقدار counter را 1 واحد افزایش می دهیم و برعکس.

برای اندازه گیری زمان اجرای برنامه از تابع clock() از کتابخانه h.time استفاده شده است، به این صورت که یکبار در ابتدا و یکبار در انتهای برنامه این تابع را صدا میزنیم و در دو متغیر begin, end ذخیره میکنیم سپس اختالف این دو عدد که برحسب کالک میباشد را تقسیم بر SEC_PER_CLOCKS میکنیم تا خروجی بر حسب ms بدست آید.

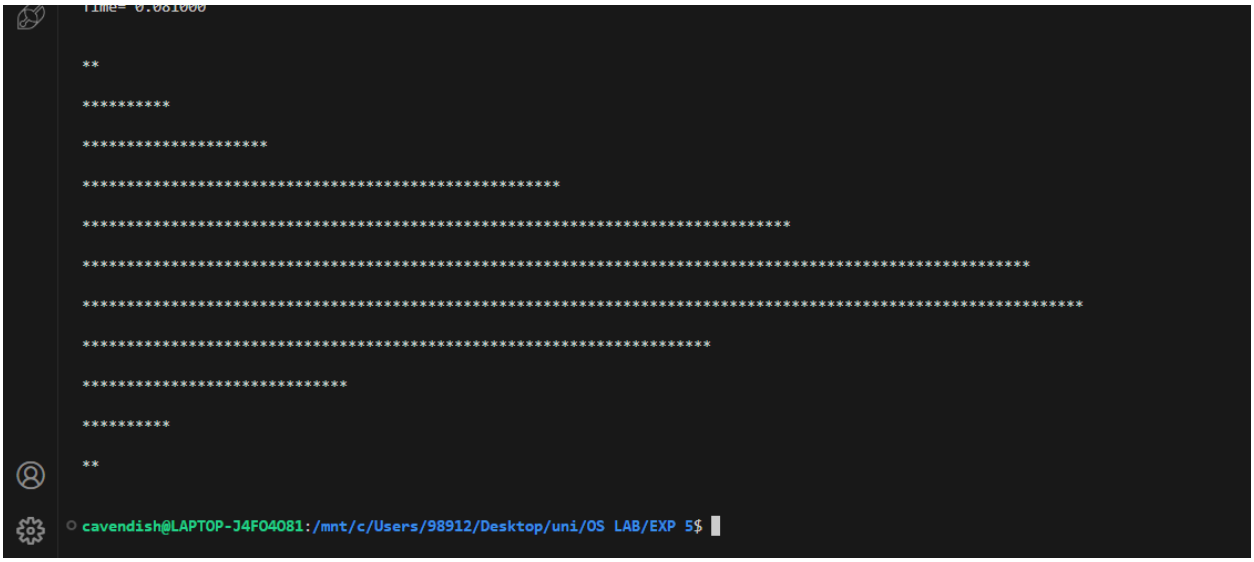
```
#include<stdio.h>
#include<stdlib.h>
#include<time.h>
void printHisogram(int *hist){
    int i,j;
    for(i = 0; i < 25; i ++){
        for (j = 0; j < (int)(hist[i]); j ++){
            printf("*");
        }
    printf("\n");
    }
}

int main(){
    clock_t begin = clock();
    srand(time(NULL));
    int hist [25] = {0};
    int n = 500000;

    for (int j = 0; j < n; j ++){
        int counter = 0;
        for (int i = 0; i < 12; i ++){
            int random_number = rand() % 100;
            if (random_number >= 49)
                counter ++;
            else
                counter --;
            }
        hist[counter + 12] ++;
    }

    clock_t end = clock();
```

```
    printf("Number of samples= %d\n", n);
    double time_spend = (double)(end - begin) / CLOCKS_PER_SEC;
    printf("Time= %f\n", time_spend);

    // printHisogram(hist);

}
```

خروجی مربوط به کد بالا در زیر مشخص است:

تعداد نمونه	۵۰۰۰	۵۰۰۰۰	۵۰۰۰۰۰
زمان اجرا (به میلی ثانیه)	۰.۷۹۶۰ ms	۷.۹۸۶۰ ms	۷۹.۹۵۳۰ ms

نمودار مربوط به این مدل برای ۵۰۰ نمونه را مشاهده می‌کنید که بصورت تابع توزیع نرمال درآمده است



بخش دوم:

در گام دوم خواسته شده که کدی که در حالت قبل با یک پردازه اجرا کرده بودیم را حال با چندین پردازه اجرا کنیم و مقایسه سرعت را انجام دهیم. به این منظور iteration ها را در 4 پردازه انجام میدهیم به این صورت که هر پردازه ¼ از تعداد کل iteration ها را انجام دهد و در آرایه به hist که با آنها به اشتراک گذاشته شده است تغییرات ایجاد کند.

ابتدا به کمک تابع shmget حافظه مشترک را ایجاد کرده و شناسه حافظه را در shegmentId میریزیم، سپس در خط دوم این حافظه مشترک را تحت عنوان hist به فضای آدرس پردازه فعلی اضافه میکنیم. سپس از دو دستور fork استفاده میکنیم تا 3 پردازه دیگر بسازیم.

سپس هر پردازه عملیات نمونه برداری را برای n/4 داده ها انجام می‌دهد و نتیجه را در hist ذخیره می‌کند. سپس هر پردازه صبر می‌کند تا کار فرزندش تمام شود. و در نهایت با توجه به مقادیر موجود، histogram را رسم میکند.

```
#include<stdio.h>
#include<stdlib.h>
#include<time.h>
#include<sys/types.h>
#include<unistd.h>
#include<sys/shm.h>
#include<sys/stat.h>
#include<sys/wait.h>

enum { ARRAY_SIZE = 25 };
int segmentId;
const int shareSize = sizeof(int) * (ARRAY_SIZE);
int *hist;

void calculate_smaple(int end){
    for (int j = 0; j < end; j ++){
        int counter = 0;
        for (int i = 0; i < 12; i ++){
            int random_number = rand() % 100;
            if (random_number >= 49)
                counter ++;
            else
                counter --;
            }
        hist[counter + 12] ++;
    }
}

void printHisogram(int *hist){
    int i,j;
    for(i = 0; i < 25; i ++){
        for (j = 0; j < (int)(hist[i] / 10); j ++)
            printf("*");
        printf("\n");
    }
}
int main(){
    clock_t begin = clock();
    segmentId = shmget(IPC_PRIVATE, shareSize, S_IRUSR | S_IWUSR);
    hist = (int *) shmat(segmentId, NULL, 0);
    srand(time(NULL));
```

```c
    int n = 5000;

    int n1 = fork();
    int n2 = fork();

    calculate_smaple(n/4);
    while(wait(NULL) != -1);

    /* parent process */
    if (n1 != 0 && n2 != 0){
        clock_t end = clock();

        int sum = 0;
        for(int i=0; i<25; i++)
            sum += hist[i];

        printf("Number of samples= %d\n", sum);
        double time_spend = (double)(end - begin) * 1000 / CLOCKS_PER_SEC;
        printf("Time= %f\n", time_spend);
        printHisogram(hist);
    }

}
```

نتایج به شکل زیر است:

تعداد نمونه	۵۰۰۰	۵۰۰۰۰	۵۰۰۰۰۰
زمان اجرا (به میلی ثانیه)	۰.۳۳۴ ms	۲.۲۴۲ ms	۲۵.۰۳۲ ms

```
cavendish@LAPTOP-J4F04081:/mnt/c/Users/98912/Desktop/uni/OS LAB/EXP 5$ ls
Q1.c  Q2.c  pic  q1  q1-500-norm  q2  report  time.txt
cavendish@LAPTOP-J4F04081:/mnt/c/Users/98912/Desktop/uni/OS LAB/EXP 5$ ./q2
Number of samples= 4758
Time= 0.771000


**

*******

***********************

**************************************************

********************************************************************************************

*******************************************************************************************************

*******************************************************************************************

********************************************************************

***************************

**********

**
```

بخش سوم:

آیا این برنامه درگیر شرایط مسابقه می شود؟چگونه؟اگر جوابتان مثبت بود راه حلی برای آن بیابید.

بله شرایط race condition فراهم است زیرا همه پردازه هایی که دسترسی به hist دارند بدون هیچ معیار و شرطی مقادیر داخل hist را تغییر می دهند و ممکن است دو پردازه در یک زمان بخواهند مقدار یکی از خانه های آرایه افزایش دهند که در این صورت ممکن است به جای 2 مرتبه افزایشی که مدنظر ما بود فقط یک مرتبه افزایش داشته باشیم.

راه حل آن است که روی قطعه کد قسمتی که روی hist عملیاتی انجام میدهیم یک lock بگذاریم به این صورت که هر پردازه قبل از انجام تغییر ابتدا اجازه دسترسی و تغییر در hist را بگیرد و این این گرفتن کلید اجازه فقط در صورتی است که پردازه دیگری در حال حاضر این کلید را در دست نداشته باشد.

بخش چهارم:

مقایسه نتایج بین ۲ قسمت اول و دوم

تعداد نمونه	5000	50000	500000
زمان اجرا سریال	۰.۷۹۶۰ ms	۷.۹۸۶۰ ms	۷۹.۹۵۳۰ ms
زمان اجرا چند پردازه ای	۰.۳۳۴ ms	۲.۲۴۲ ms	۲۵.۰۳۲ ms
تسریع	۰.۴۶۲ ms	۵.۷۴۴ ms	۵۴.۹۲۱ ms
تسریع – درصد	%۵۸.۰۴	%۷۱.۹۳	%۶۸.۶۹