آزمایش۵)

امید خباز قانع ۹۹۳۱۰۱۸– محمدمهدی نظری۹۹۳۱۰۶۱

قسمت اول)

به این صورت که برای محاسبه زمان طول کشیده برای اجرای برنامه را به این صورت حساب میکنیم که ساعت شروع برنامه start و پایان end را با استفاده از ()clock میگیریم و duration برنامه را حساب میکنیم.

بقیه هم طبق الگوریتمی که گفته شده است و با توجه به تعداد نمونه هایی که داریم size مربوط به آرایه hist را تغییر و همچنین تعداد تکرار حلقه بیرونی را نیز تغییر داده و در هر مورد duration را حساب میکنیم.

Demu > C RandNumber.c

```c
#include <stdio.h>
#include<stdlib.h>
#include<time.h>
int main(){
    clock_t start, end;
    double duration;
    start = clock();

    int hist[5000];
    for(int i = 0; i<5000; i++)
    {
        int counter = 0;
        for(int j = 0; j<12; j++)
        {
            int r1 = rand();
            float r = ((float)rand()/(float)(RAND_MAX)) * 100;
            // printf("%f\n",r);
            if(r >= 49)
            {
                counter++;
            }
            else
            {
                counter--;
            }
        }
        hist[i] = counter;
        // printf("%d\n",counter);
    }
    end = clock();
    duration = ((double)(end - start))/CLOCKS_PER_SEC;

    printf("time spent: %f \n", duration);
    return 0;
}
```

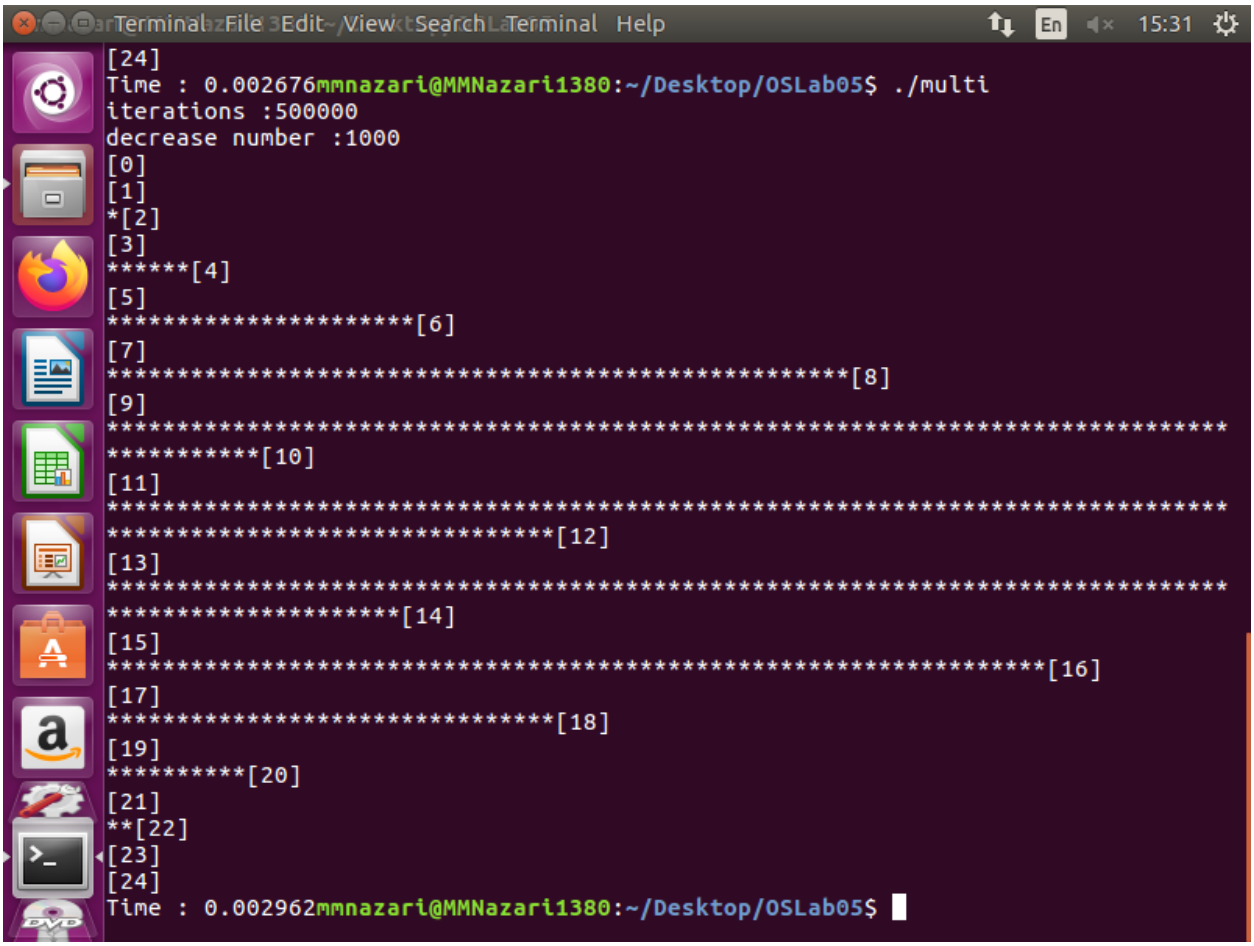
خروجی اول ۵۰۰۰۰۰ و دومی ۵۰۰۰۰ و سومی ۵۰۰۰ است:

```
root@7e13ce0f06d0:/Demu# gcc RandNumber.c -o RandNumber
root@7e13ce0f06d0:/Demu# ./RandNumber
time spent: 0.193598
root@7e13ce0f06d0:/Demu# gcc RandNumber.c -o RandNumber
root@7e13ce0f06d0:/Demu# ./RandNumber
time spent: 0.024537
root@7e13ce0f06d0:/Demu# gcc RandNumber.c -o RandNumber
root@7e13ce0f06d0:/Demu# ./RandNumber
time spent: 0.003318
root@7e13ce0f06d0:/Demu#
```

بخش ۲)

```
mmnazari@MMNazari1380: ~/Desktop/OSLab05
mmnazari@MMNazari1380:~/Desktop/OSLab05$ gcc multi.c -o multi
mmnazari@MMNazari1380:~/Desktop/OSLab05$ ./multi
iterations :5000
decrease number :10
[0]
[1]
[2]
[3]
*****[4]
[5]
**********************[6]
[7]
*********************************************[8]
[9]
*****************************************************************************
********************[10]
[11]
*****************************************************************************
********************************[12]
[13]
*****************************************************************************
********************[14]
[15]
*************************************************************[16]
[17]
********************************[18]
[19]
*********[20]
[21]
[22]
[23]
[24]
Time : 0.003680mmnazari@MMNazari1380:~/Desktop/OSLab05$
```

```
Terminal File Edit View Search Terminal Help
[24]
mmnazari@MMNazari1380:~/Desktop/OSLab05$ ./multi
iterations :50000
decrease number :100
[0]
[1]
*[2]
[3]
******[4]
[5]
**********************[6]
[7]
*********************************************[8]
[9]
*****************************************************************************
************[10]
[11]
*****************************************************************************
********************************[12]
[13]
*****************************************************************************
**********************[14]
[15]
*************************************************************[16]
[17]
*****************************[18]
[19]
**********[20]
[21]
**[22]
[23]
[24]
Time : 0.002676mmnazari@MMNazari1380:~/Desktop/OSLab05$
```



۵۰۰۰۰۰	۵۰۰۰۰	۵۰۰۰	تعداد نمونه
2.96 ms	2.67 ms	3.68 ms	زمان اجرا

در این حالت ۵ پردازه ساخته شده و تعداد عملیات های ورودی را تقسیم بر ۵ کردم که می شود تعداد عملیاتی که یک پردازه باید انجام بدهد و به این شکل هر پردازه ۱/۵ تعداد کل عملیات ها را انجام می دهد . (هر پردازه بعد از اینکه هر پردازه محاسبات را انجام داد تمام می شود و در نهایت پردازه mainرسم نمودار را انجام می دهد

برای ارتباط بین پردازه ها هم از روش حافظه مشترک استفاده شده است یعنی یک ارایه ۲۵ تایی در یک حافظه مشترک تعریف شده است که پردازه های مختلف می توانند در آن مقادیر را بنویسند.

برای اینکه از نظر شکلی بتوان در یک صفحه دید و مناسب باشد یک عدد را ورودی گرفتم و تعداد ستاره های چاپی را بر آن تقسیم کردم که تعداد کمی کمتر بشود که از نظر ظاهری بهتر باشد.

بخش ۳)

حالت مسابقه به حالتی گفته می شود که مثل یک مقدار بین دو پردازه مختلف مشترک است و هردو می توانند آن را بخوانند و بنویسند .

حال با توجه به اینکه ترتیب ثابتی برای اجرای پردازه های تولید شده وجود ندارد پس هر کدام می توانند زودتر یا دیرتر مقادیر خود را بنویسند و یا بخوانند و در نتیجه خروجی برای چند بار اجرا لزوما یکی نیست و نتایج مختلفی (با توجه به ترتیب اجرای پردازه ها) ممکن است به وجود بیاید . به این وضعیت حالت مسابقه می گویند

. در این مثال حالت مسابقه وجود دارد . مثل در این مثال ممکن است مقدار یک خانه زیاد شود و بعد کم بشود یا دو بار زیاد شود و منظور این است که در hist[counter +12]حالت race ممکن است وجود داشته باشد . البته چون در این مثال کل ما در حال استفاده از اعداد تصادفی هستیم این حالت مسابقه تاثیر نامطلوبی در نتیجه کار ما ندارد . برای جلوگیری از این روش می توان از سمافور استفاده کرد به این شکل که باید برای هر یک از ۲۵ خانه ارایه یک سمافور تعریف کرد که متغیری از جنسsem_t است که مقدار آن باید ۱ باشد تا وقتی که یک پردازه در حال نوشتن یا خواندن است و با آن خانه کار دارد مقدار سمافور ۰ می شود و این باعث می شود که دیگران نتوانند از آن خانه استفاده بکنند (وقتی کار تمام می شود مقدار دوباره ۱ می شود تا به دیگران اجازه استفاده داده بشود) و در ادامه به کمک دستور های زیر می توانیم از حالت مسابقه جلوگیری کنیم . (به کمکsem_init() می توانیم مقایر اولیه را برای سمافور ها قرار دهیم)

```
sem_wait(&sems[counter + 12]);

hist[counter + 12]+=1;

sem_post(&sems[counter + 12]);
```

به کمک دو دستور اضافه شده در بالا می توان جلوی حالت مسابقه را گرفت.

بخش ۴)

تعداد نمونه	5000	50000	500000
افزایش سرعت (اختلاف زمان)	-0.37 ms	+21.86 ms	+190.59 ms