موژان میرجلیلی - ۹۸۳۱۱۴۰

۵۰۰۰

```
moujanmirjalili@ubuntu:~/Desktop/lab5$ ./serial
index: -12, number:      1 |
index: -11, number:      0 |
index: -10, number:     15 |
index:  -9, number:      0 |
index:  -8, number:     69 |*
index:  -7, number:      0 |
index:  -6, number:    268 |*****
index:  -5, number:      0 |
index:  -4, number:    617 |***********
index:  -3, number:      0 |
index:  -2, number:    934 |*****************
index:  -1, number:      0 |
index:   0, number:   1142 |**********************
index:   1, number:      0 |
index:   2, number:    948 |*****************
index:   3, number:      0 |
index:   4, number:    628 |***********
index:   5, number:      0 |
index:   6, number:    271 |*****
index:   7, number:      0 |
index:   8, number:     93 |*
index:   9, number:      0 |
index:  10, number:     14 |
index:  11, number:      0 |
index:  12, number:      0 |

Total time: 0.001520
```

۵۰۰۰۰

```
moujanmirjalili@ubuntu:~/Desktop/lab5$ ./serial
index: -12, number:      9 |
index: -11, number:      0 |
index: -10, number:    124 |
index:  -9, number:      0 |
index:  -8, number:    723 |*
index:  -7, number:      0 |
index:  -6, number:   2609 |*****
index:  -5, number:      0 |
index:  -4, number:   5883 |**********
index:  -3, number:      0 |
index:  -2, number:   9426 |*****************
index:  -1, number:      0 |
index:   0, number:  11417 |**********************
index:   1, number:      0 |
index:   2, number:   9658 |******************
index:   3, number:      0 |
index:   4, number:   6298 |***********
index:   5, number:      0 |
index:   6, number:   2831 |*****
index:   7, number:      0 |
index:   8, number:    837 |*
index:   9, number:      0 |
index:  10, number:    176 |
index:  11, number:      0 |
index:  12, number:      9 |

Total time: 0.013819
```

```
moujanmirjalili@ubuntu:~/Desktop/lab5$ ./serial
index: -12, number:    112 |
index: -11, number:      0 |
index: -10, number:   1324 |
index:  -9, number:      0 |
index:  -8, number:   7564 |*
index:  -7, number:      0 |
index:  -6, number:  25569 |*****
index:  -5, number:      0 |
index:  -4, number:  58249 |**********
index:  -3, number:      0 |
index:  -2, number:  95133 |******************
index:  -1, number:      0 |
index:   0, number: 112265 |**********************
index:   1, number:      0 |
index:   2, number:  98256 |******************
index:   3, number:      0 |
index:   4, number:  62694 |************
index:   5, number:      0 |
index:   6, number:  28222 |*****
index:   7, number:      0 |
index:   8, number:   8825 |*
index:   9, number:      0 |
index:  10, number:   1655 |
index:  11, number:      0 |
index:  12, number:    132 |

Total time: 0.084450
```

تعداد نمونه	۵۰۰۰	۵۰۰۰۰	۵۰۰۰۰۰
زمان اجرا	۰,۰۰۱۵۲۰	۰,۰۱۳۸۱۹	۰,۰۸۴۴۵۰

توضیح کد:

ابتدا NUM را به ازای مقدار دلخواه (۵۰۰۰، ۵۰۰۰۰، ۵۰۰۰۰۰) define کرده. تابع calcCounter برای محاسبه ۱۲ مقدار رندوم طبق توضیح دستورکار نوشته شده است و مقدار counter را برمی‌گرداند.

تابع printHist مشابه دستورکار برای چاپ نتایج نوشته شده تا بتوانیم خروجی را به صورت نمودار منحی نرمال ترسیم کنیم، تنها تفاوتی که با دستورکار دارد این است که تعداد ستاره‌هایی که باید برای هر ردیف چاپ شود را بر ۱۰۰ تقسیم کردیم تا نتایج در خروجی در یک سطر قابل مشاهده باشند.

Srand(tame(0)) برای عدد رندوم، startTime برای نگه داشتن زمان شروع و endTime در انتها برای زمان پایان تعریف شده است.

ابتدا آرایه را مقداردهی اولیه (۰) می‌کنیم، سپس به تعداد نمونه‌ها تابع را صدا زده و مثل دستورکار مقدار خانه‌های آرایه را تغییر می‌دهیم. در آخر خروجی‌ها را چاپ می‌کنیم و در آخر با به دست آوردن اختلاف زمان شروع و پایان و تقسیم آن‌ها به CLOCKS_PER_SEC زمانی اجرا را بر حسب ثانیه محاسبه و چاپ می‌کنیم.

Open | serial.c ~/Desktop/lab5 | Save

```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

#define NUM 500000

int calcCounter(){
    int counter = 0;
        for(int i=0; i<12; i++){
            int random = rand()%101;
                //printf("%d ", random);
            if(random>49){
                counter++;
            }else{
                counter--;
            }
        }
        //printf("\n");
        return counter;
}

void printHist(int *hist){
    for (int i = 0; i < 25; i++) {
                printf("index: %3d, number: %6d |",i-12,hist[i]);
        for (int j = 0; j < hist[i]/(NUM/100); j++) {
            printf("*");
        }
        printf("\n");
    }
}

int main(void) {
        srand(time(0));
    clock_t startTime = clock();
        int hist[25];
        for(int i = 0;i<25;i++){
            hist[i]=0;
        }

        for (int i = 0; i < NUM; i++) {
            int counter = calcCounter();
                //printf("%d \n", counter);
            hist[12+counter]++;
        }

        printHist(hist);

    clock_t endTime = clock();
    printf("\nTotal time: %f \n", ((double)(endTime - startTime)/-
CLOCKS_PER_SEC));
        return 0;
}
```

```
moujanmirjalili@ubuntu:~/Desktop/lab5$ ./parallel
5000

Time Spent 0.001119
index: -12, number:      0 |
index: -11, number:      0 |
index: -10, number:     10 |
index:  -9, number:      0 |
index:  -8, number:     50 |*
index:  -7, number:      0 |
index:  -6, number:    160 |***
index:  -5, number:      0 |
index:  -4, number:    560 |***********
index:  -3, number:      0 |
index:  -2, number:    899 |*****************
index:  -1, number:      0 |
index:   0, number:   1164 |***********************
index:   1, number:      0 |
index:   2, number:   1026 |********************
index:   3, number:      0 |
index:   4, number:    640 |************
index:   5, number:      0 |
index:   6, number:    350 |*******
index:   7, number:      0 |
index:   8, number:    120 |**
index:   9, number:      0 |
index:  10, number:     10 |
index:  11, number:      0 |
index:  12, number:      0 |
moujanmirjalili@ubuntu:~/Desktop/lab5$
```

```
moujanmirjalili@ubuntu:~/Desktop/lab5$ ./parallel
50000

Time Spent 0.001019
index: -12, number:      0 |
index: -11, number:      0 |
index: -10, number:    100 |
index:  -9, number:      0 |
index:  -8, number:    570 |*
index:  -7, number:      0 |
index:  -6, number:   2373 |****
index:  -5, number:      0 |
index:  -4, number:   4792 |*********
index:  -3, number:      0 |
index:  -2, number:   8660 |*****************
index:  -1, number:      0 |
index:   0, number:  11469 |**********************
index:   1, number:      0 |
index:   2, number:  10001 |********************
index:   3, number:      0 |
index:   4, number:   6424 |************
index:   5, number:      0 |
index:   6, number:   3310 |******
index:   7, number:      0 |
index:   8, number:   1197 |**
index:   9, number:      0 |
index:  10, number:    179 |
index:  11, number:      0 |
index:  12, number:     10 |
moujanmirjalili@ubuntu:~/Desktop/lab5$
```

```
moujanmirjalili@ubuntu:~/Desktop/lab5$ ./parallel
500000

Time Spent 0.001294
index: -12, number:    170 |
index: -11, number:      0 |
index: -10, number:   1240 |
index:  -9, number:      0 |
index:  -8, number:   6041 |*
index:  -7, number:      0 |
index:  -6, number:  23176 |****
index:  -5, number:      0 |
index:  -4, number:  53859 |**********
index:  -3, number:      0 |
index:  -2, number:  58865 |***********
index:  -1, number:      0 |
index:   0, number: 108010 |**********************
index:   1, number:      0 |
index:   2, number:  99384 |*******************
index:   3, number:      0 |
index:   4, number:  56793 |***********
index:   5, number:      0 |
index:   6, number:  31641 |******
index:   7, number:      0 |
index:   8, number:   9683 |*
index:   9, number:      0 |
index:  10, number:   2020 |
index:  11, number:      0 |
index:  12, number:    140 |
moujanmirjalili@ubuntu:~/Desktop/lab5$
```

تعداد نمونه	۵۰۰۰	۵۰۰۰۰	۵۰۰۰۰۰
زمان اجرا	۰,۰۰۱۱۱۹	۰,۰۰۱۰۱۹	۰,۰۰۱۲۹۴

توضیح کد:

در این قسمت از مفهوم fork استفاده کردیم و id والد را در ریشه root_pid قرار دادیم. ریشه در ابتدا تعداد دلخواهی فرزند تولید می‌کند و هر فرزند {(تعداد فرزندها)/ (تعداد نمونه‌ها)}، عملیات انجام می‌دهد.
در اینجا از مفهوم shared memory استفاده کردیم و همگی به صورت اشتراکی با این حافظه کار می‌کنند.

Open | parallel.c ~/Desktop/lab5 | Save

```c
#include <stdlib.h>
#include <unistd.h>
#include <sys/wait.h>
#include <sys/types.h>
#include <string.h>
#include <pthread.h>
#include <sys/ipc.h>
#include <sys/shm.h>
#include <time.h>
#include <stdio.h>

int const num = 10;
int id;
int root_pid;
int pid;

typedef struct{
    int histData[25];
} Hist;

Hist *hist;

int main()
{
    int iteration;

    scanf("%d", &iteration);

    clock_t start = clock();

    srand(time(0));

    id = shmget(IPC_PRIVATE, sizeof(Hist), IPC_CREAT | 0666);

    root_pid = getpid();

    for (int i = 0; i < num; i++)
    {
        if (getpid() == root_pid)
            pid = fork();
        else
            break;
    }

    hist = (Hist *)shmat(id, NULL, 0);

    if (getpid() == root_pid)
    {
        for (int i = 0; i < num; i++)
        {
            wait(NULL);
        }
    }
    else
    {

        int count, random;
        for (int i = 0; i < iteration / num; i++)
        {
            count = 0;
            for (int j = 0; j < 12; j++)
            {
                random = rand() % 101;
                if (random >= 49)
                {
                    count++;
                }
                else
                {
                    count--;
                }
            }
            hist->histData[count + 12]++;
        }

        exit(0);
    }

    clock_t end = clock();
    double time_spent = (double)(end - start)/CLOCKS_PER_SEC;
    printf("\nTime Spent %lf\n", time_spent);

      for (int i = 0; i < 25; i++){
         printf("index: %3d, number: %6d |", i - 12, hist->histData[i]);
          for(int j = 0; j < (int)(hist->histData[i]/(iteration/100)); j++) {
           printf("*");
        }
         printf("\n");
        }

    return 0;
}
```

۳

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

جوابتان مثبت بود راه حلی برای آن بیابید.

بله، در شرایطی که دو پردازه فرزند بخواهند مقدار یک خانه از آرایه که در حافظه مشترک (Critical Section) است را تغییر دهند و دسترسی داشته باشند ممکن است شرایط مسابقه پیش آید و مقدار آن خانه از آرایه اشتباه تغییر کند. برای حل این مشکل از Semaphore استفاده می‌کنیم.

سمافور می‌تواند به دو صورت دودویی (که تنها دو مقدار صحیح و غلط را دارا است) یا شمارنده اعداد صحیح باشد. از سمافور برای جلوگیری از ایجاد شرایط مسابقه میان فرایندها استفاده می‌گردد. به این ترتیب، اطمینان حاصل می‌شود که در هر لحظه تنها یک فرایند به منبع مشترک دسترسی دارد و می‌تواند از آن بخواند یا بنویسد.

شمارش سمافورها با دو عملیات همراه است:

Wait(): مقدار سمافور را یک واحد کاهش داده و یک واحد از منبع اشتراکی را مصرف می‌کند. اگر در هنگام کاهش، مقدار منفی شد، پروسه‌ای که wait() را اجرا کرده بلوکه می‌شود و در انتهای صف سمافور قرار می‌گیرد تا منابع توسط پروسه‌های دیگر آزاد شوند.

Signal(): مقدار سمافور را یک واحد افزایش می‌دهد. پس از افزایش دادن، اگر مقدار قبل سمافور منفی باشد (به این معنی که در حال حاضر پروسه‌هایی در صف سمافور منتظر دریافت منبع هستند)، یکی از پروسه‌ها از صف آماده وارد صف اجرا می‌شود و منبع آزاد شده را در اختیار می‌گیرد.

۴

تعداد نمونه	۵۰۰۰	۵۰۰۰۰	۵۰۰۰۰۰
افزایش سرعت	۷۳,۶۱	۷,۳۷	۱,۵۳

با توجه به افزایش سرعت می‌بینیم که موازی‌سازی همواره باعث بهبود سرعت شده است.