به نام خدا

محمد جواد زندیه

9831032

آزمایش 5 سیستم عامل

---

گام یک:

ابتدا کد برنامه اي که در تعریف مسئله شرح داده شد را در حالت سریال بنویسید و زمان اجرا شدن برنامه خود را در جدول زیر گزارش دهید.

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

int main(int argc, char const *argv[]){
    int hist[25];
    for(int i = 0; i < 25; i++)
        hist[i] = 0;
    int iter = 500000;

    double time_spent = 0.0;
    clock_t begin = clock();

    srand(time(NULL));
    for(int j = 0; j < iter; j++){
        int counter = 0;
        for(int i = 0; i < 12; i++){
            int r = rand() % 100;
            if(r >= 49)
                counter++;
            else
                counter--;
        }
        hist[counter + 12]++;
    }
```

```c
    clock_t end = clock();
    time_spent += (double)(end - begin) / CLOCKS_PER_SEC;
    printf("The elapsed time for %d iteration is %f seconds.\n", iter ,time_spent);

    printf("Normal Distribution history: ");
    for(int j = 0; j < 25; j++)
        printf("%d ", hist[j]);
    printf("\n");

    return 0;
}
```

خروجی به ازای تکرار های مختلف:

```
javad@javad-HP-350-G1:~/Desktop/OSLab/project5$ gcc NormalDistribution.c -o NormalDistribution
javad@javad-HP-350-G1:~/Desktop/OSLab/project5$ ./NormalDistribution
The elapsed time for 5000 iteration is 0.001278 seconds.
Normal Distribution history: 3 0 11 0 74 0 232 0 580 0 904 0 1081 0 1027 0 662 0 308 0 99 0 16 0 3
javad@javad-HP-350-G1:~/Desktop/OSLab/project5$ gcc NormalDistribution.c -o NormalDistribution
javad@javad-HP-350-G1:~/Desktop/OSLab/project5$ ./NormalDistribution
The elapsed time for 50000 iteration is 0.017752 seconds.
Normal Distribution history: 22 0 114 0 651 0 2422 0 5576 0 9215 0 11089 0 10140 0 6536 0 3046 0 995 0 181 0 13
javad@javad-HP-350-G1:~/Desktop/OSLab/project5$ gcc NormalDistribution.c -o NormalDistribution
javad@javad-HP-350-G1:~/Desktop/OSLab/project5$ ./NormalDistribution
The elapsed time for 500000 iteration is 0.136035 seconds.
Normal Distribution history: 91 0 1234 0 6883 0 23715 0 55536 0 92656 0 112448 0 100323 0 65548 0 30122 0 9578 0 1703 0 163
javad@javad-HP-350-G1:~/Desktop/OSLab/project5$
```

| تعداد نمونه | 5000 | 50000 | 500000 |
|---|---|---|---|
| زمان اجرا | 0.001278 s | 0.017752 s | 0.136035 s |

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.

شکل histogram به ازای 500 نمونه هم رسم شده است که نشان می دهید به درستی به شکل زنگوله ای توزیع نرمال رسیده ایم.

```
javad@javad-HP-350-G1:~/Desktop/OSLab/project5$ ./NormalDistribution
The elapsed time for 500 iteration is 0.000220 seconds.


**

*****

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

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

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

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

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

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

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

*******

**

Normal Distribution history: 0 0 2 0 5 0 24 0 58 0 96 0 105 0 98 0 70 0 33 0 7 0 2 0 0
```

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 shm_hist که با آنها به اشتراک گذاشته شده است تغییرات ایجاد کند.

NormalDistribution_thread.c > main(int, char const * [])

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


void printHistogram(int* hist){
    int i, j;
    for(i = 0; i < 25; i++){
        for(j = 0; j < hist[i]; j++)
            printf("*");
        printf("\n");
    }
}
```

```c
void writeResualt(clock_t begin, clock_t end, int* hist, int iter){
    double time_spent = 0.0;
    time_spent += (double)(end - begin) / CLOCKS_PER_SEC;
    printf("The elapsed time for %d iteration is %f seconds.\n", iter ,time_spent);

    // printHistogram(hist);
    printf("Normal Distribution history: ");
    for(int j = 0; j < 25; j++)
        printf("%d ", hist[j]);
    printf("\n");
}

void do_iteration(int iter, int* shm_hist){
    for(int j = 0; j < iter; j++){
        int counter = 0;
        for(int i = 0; i < 12; i++){
            int r = rand() % 100;
            if(r >= 49)
                counter++;
            else
                counter--;
        }
        shm_hist[counter + 12]++;
    }
}
```

```c
int main(int argc, char const *argv[]){
    // shared memory id
    int shm_id = shmget(IPC_PRIVATE, 20, IPC_CREAT| SHM_R | SHM_W);
    // shared memory attach
    int* shm_hist = (int *)shmat(shm_id, 0, 0);
    // initialize shared memory with 0
    for(int i = 0; i < 25; i++)
        shm_hist[i] = 0;

    // set number of iteration for all
    int iter = 5000;
    // set number of itteration for each children
    // we have 4 processes thus we split iteration into 4 parts
    int child_iter = iter/4;

    // for generating random number we have to call it one in our code
    srand(time(NULL));

    // start of normal distribution computation
    clock_t begin = clock();
```

```c
    // four process
    if(fork()){ //child1
        do_iteration(child_iter, shm_hist);
        exit(EXIT_SUCCESS);
    }else{
        if(fork()){ //child2
            do_iteration(child_iter, shm_hist);
            exit(EXIT_SUCCESS);
        }else{
            if(fork()){ //child3
                do_iteration(child_iter, shm_hist);
                exit(EXIT_SUCCESS);
            }else{ //parent
                int* shm_hist = (int *)shmat(shm_id, 0, 0);
                do_iteration(child_iter, shm_hist);
                wait(NULL);
            }
        }
    }

    clock_t end = clock();

    writeResualt(begin, end, shm_hist, iter);
    shmctl(shm_id, IPC_RMID, NULL);

    return 0;
}
```

با استفاده از fork سه فرزند برای parent ایجاد کردیم و به هر فرزند وظیفه ¼ از کار را دادیم.

خروجی:

```
javad@javad-HP-350-G1:~/Desktop/OSLab/project5$ gcc NormalDistribution_thread.c -o NormalDistribution_thread
javad@javad-HP-350-G1:~/Desktop/OSLab/project5$ ./NormalDistribution_thread
javad@javad-HP-350-G1:~/Desktop/OSLab/project5$ The elapsed time for 5000000 iteration is 0.474720 seconds.
Normal Distribution history: 819 0 11890 0 68498 0 232331 0 533988 0 868801 0 1031718 0 935137 0 626395 0 294103 0 92343 0 17452 0 1443

javad@javad-HP-350-G1:~/Desktop/OSLab/project5$ gcc NormalDistribution_thread.c -o NormalDistribution_thread
javad@javad-HP-350-G1:~/Desktop/OSLab/project5$ ./NormalDistribution_thread
javad@javad-HP-350-G1:~/Desktop/OSLab/project5$ The elapsed time for 500000 iteration is 0.043554 seconds.
Normal Distribution history: 64 0 1151 0 7039 0 23618 0 55208 0 90076 0 108149 0 97495 0 52668 0 29933 0 9680 0 1888 0 124

javad@javad-HP-350-G1:~/Desktop/OSLab/project5$ gcc NormalDistribution_thread.c -o NormalDistribution_thread
javad@javad-HP-350-G1:~/Desktop/OSLab/project5$ ./NormalDistribution_thread
javad@javad-HP-350-G1:~/Desktop/OSLab/project5$ The elapsed time for 50000 iteration is 0.003218 seconds.
Normal Distribution history: 12 0 156 0 635 0 2426 0 5388 0 8681 0 11036 0 9777 0 6387 0 3077 0 929 0 124 0 32

javad@javad-HP-350-G1:~/Desktop/OSLab/project5$ gcc NormalDistribution_thread.c -o NormalDistribution_thread
javad@javad-HP-350-G1:~/Desktop/OSLab/project5$ ./NormalDistribution_thread
javad@javad-HP-350-G1:~/Desktop/OSLab/project5$ The elapsed time for 5000 iteration is 0.000048 seconds.
Normal Distribution history: 0 0 12 0 68 0 244 0 546 0 898 0 1098 0 938 0 635 0 327 0 80 0 12 0 0

javad@javad-HP-350-G1:~/Desktop/OSLab/project5$
```

| تعداد نمونه | 5000 | 50000 | 500000 |
|---|---|---|---|
| زمان اجرا | 0.000048 s | 0.003218 s | 0.043554 s |

---

۳. ایا این برنامه درگیر شرایط مسابقه می شود؟چگونه؟اگر جوابتان مثبت بود راه حلی براي آن بیابید.

بله شرایط race condition فراهم است زیرا همه پردازه هایی که دسترسی به shm_hist دارند بدون هیچ معیار و شرطی مقادیر داخل shm_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 Shm_hist عملیاتی انجام میدهیم یک lock بگذاریم به این صورت که هر پردازه قبل از انجام تغییر ابتدا اجازه دسترسی و تغییر در shm_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.

در تصویر زیر مشاهده می شود که اجرای پردازه ها هم زمان می باشد زیرا مثلا end پردازه 1 بعد از start پردازه 2 است و ... که این موجب race condition می شود چون هر دو روی آرایه یکسان کار می کنند.

```
javad@javad-HP-350-G1:~/Desktop/OSLab/project5$ gcc NormalDistribution_thread_withoutRace.c -o NormalDistribution_thread_withoutRace
javad@javad-HP-350-G1:~/Desktop/OSLab/project5$ ./NormalDistribution_thread_withoutRace
s1
s2
s3
e1
e2
javad@javad-HP-350-G1:~/Desktop/OSLab/project5$ e3
sp
ep
The elapsed time for 5000 iteration is -0.000588 seconds.
Normal Distribution history: 0 0 12 0 99 0 242 0 502 0 920 0 1168 0 938 0 643 0 274 0 100 0 12 0 0

javad@javad-HP-350-G1:~/Desktop/OSLab/project5$
```

---

میزان تسریع عملیات با استفاده از روش چند پردازه ای نسبت به حالت سریال:

| تعداد نمونه | 5000 | 50000 | 500000 |
|---|---|---|---|
| زمان اجرا سریال | 0.001278 s | 0.017752 s | 0.136035 s |
| زمان اجرا چند پردازه ای | 0.000048 s | 0.003218 s | 0.043554 s |
| تسریع | 26.625 | 5.5 | 3.12 |