

ازمایش 5

بدون استفاده از fork:

ابتدا کتابخانه های لازم را فراخوانی میکنیم

سپس یک آرایه 25 تایی تعریف میکنیم

سپس وارد حلقه for میشویم و مطابق با دستور کار کاره لازم را انجام میدهیم

```
for (int i = 0; i<number ; ++i)
{
    int counter=12;
    for (int i = 0; i < 12; ++i)
    {
        int random = rand()%100;

        if(random>=49)
            counter+=1;
        else
            counter-=1;
    }
    hist[counter] +=1;
}
```

سپس با یک حلقه ساده ان را چاپ میکنیم

دلیل کم کردن از 12 به دلیل این است که خانه 0 اندیس 12 را دارد برای سادگی کار

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

قسمت دوم نوشتن کد با استفاده از fork:

ابتدا کتابخانه های لازم را فراخوانی میکنیم

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

#include <math.h>
```

سپس یک shared memory میسازیم (مطابق با کار هایی که در دستور کار قبلی انجام میدهم)

```
int pId;
    int segment_id;

    int* shared_memory;

    int shared_segment_size = 25*sizeof(int);

    key_t key = ftok("shmfile",65);

    segment_id = shmget(key, shared_segment_size,0666|IPC_CREAT);
    int* hist = (int*)shmat(segment_id,NULL,0)

    سپس از hist را به ارایه 25 تایی تبدیل میکنیم و هر قسمت ان را 0 میگذاریم
for (int i = 0; i < 25; ++i)
{
    hist[i]=0;
}
```

سپس واردیم حلقه 100 تایی میشویم که در هر مرحله با دستور fork در هر بار اجرا یک پدازه فرزند ایجاد میکنیم و دراز انجا که مقدار بازگردانده شده برای فرزند 0 است فرایند فرزند وارد شرط اول میشود و پردازش فرزند از ابتدای شرط اول شروع به اجرا شدن میکند هر فرایند فرزند یک صدم از کار ها را قبول میکند و شروع به انجام ان ها میکند و در پایان با دستور exit(0) خود را تمام میکند و نتیجه نهایی را در shared memory ذخیره میکند

اگر در ایجاد فرایند فرزند دچار مشکل یا خطا شود سیستم عامل در شرط دوم با توجه به اینکه خروجی fork منفی است خطا چاپ میکند و یک بار دیگر این مرحله را تکرار میکند تا فرزندان از 100 تا کمتر نشوند

در پایین این حلقه فرایند پدر منتظر می ماند تا همه بچه ها برسند و به کار خود ادامه دهد که این کار را باه شکل زیر انجام

```
while((wpid=wait(&status))>0);
```

منفی میشود wait این کار را با توجه به اینکه اگر فرزندی وجود نداشته باشد مقدار خروجی

```

میدهدfor (int j = 0; j < 100; ++j)
{
    pId = fork();
    if(pId==0){

        for (int i = 0; i<number/100 ; ++i)
        {
            int counter=12;
            for (int i = 0; i < 12; ++i)
            {
                int random = rand()%100;

                if(random>=49)
                    counter+=1;
                else
                    counter-=1;
            }

            hist[counter] +=1;

        }
        //printf("%s\n","here" );
        exit(0);
    }
    if(pId<0)
        {printf("%s\n", "error"); j--;}
}
int status=0;
pid_t wpid;
while((wpid=wait(&status))>0);

```

در پایان نیز مانند قسمت قبل ان ها را و نتیجه را چاپ میکنیم

:Race condition

در اینجا چون از shared memory استفاده کرده ایم هنگامی که روی ان 1 واحد اضافه کنیم ممکن ست همزمان پردازش دیگری نیز در همان لحظه دقیقاً در حال نوشتن اضافه کردخ 1 واحد باشد یا هر چند تا پردازش دیگر به همین دلیل به جای اینکه تعداد به تعداد پردازش ها اضافه شود به مقدار درون shared memory فقط یک واحد اضافه میشود و بنابر این باعث میشود از مقدار داده شده اولیه کمتر شود که برای حل این شرایط با توجه به مباحث درس میتوان از سمافور استفاده کرد

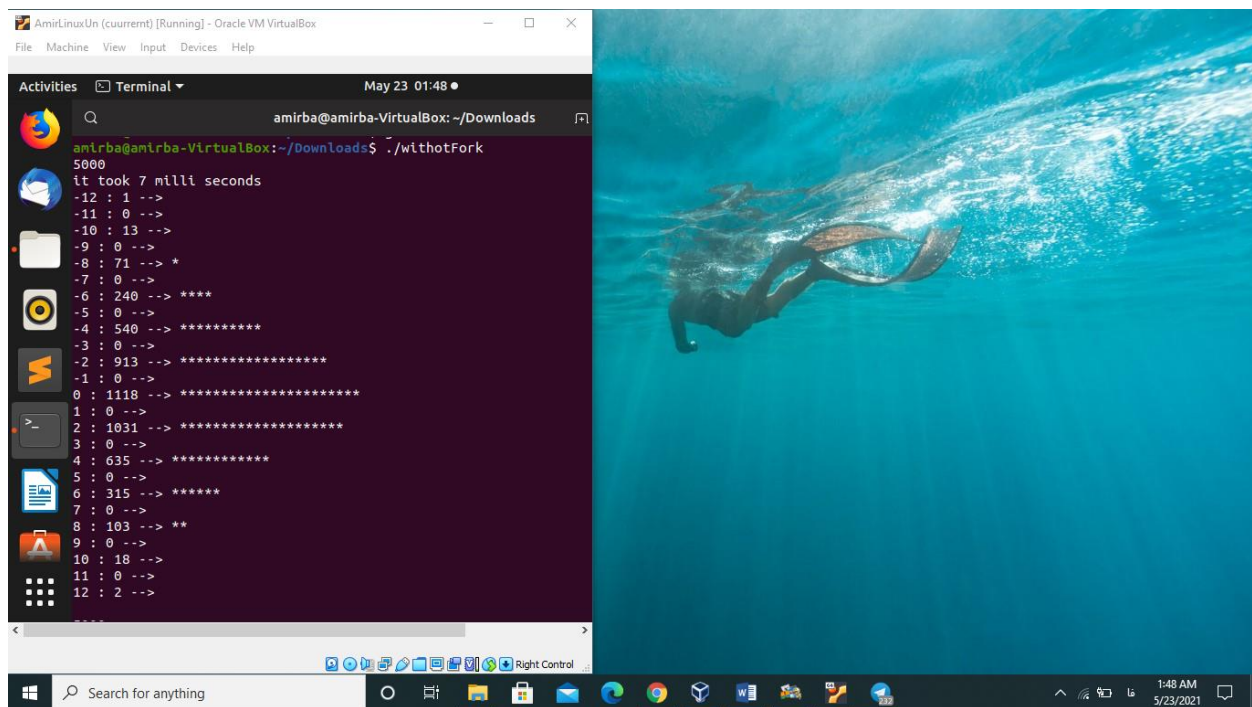
خروجی ها:

بدون فورک :

7 میلی ثانیه = 5000

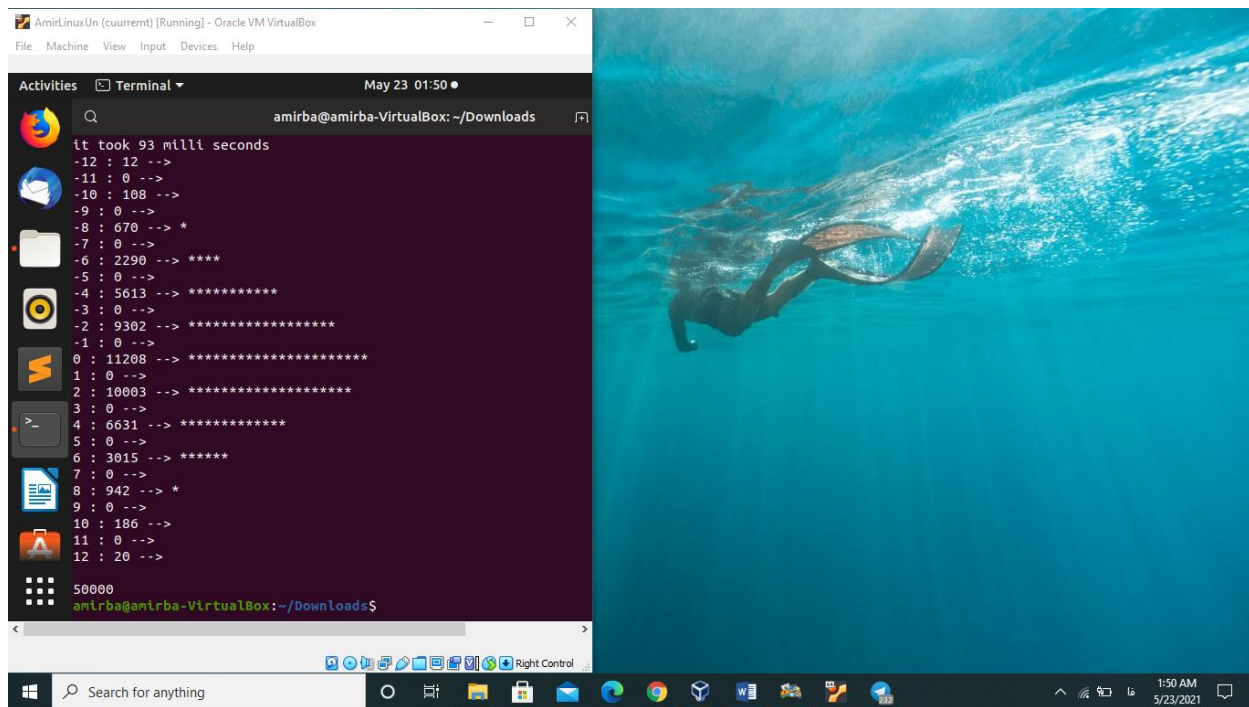
70 میای ثانیه = 50000

701 میلی ثانیه = 500000



The screenshot shows a Windows 10 desktop with a taskbar at the bottom. A virtual machine window titled "AmirLinuxUn (current) [Running] - Oracle VM VirtualBox" is open. Inside the VM, a terminal window shows the following output:

```
amirba@amirba-VirtualBox: ~/Downloads
amirba@amirba-VirtualBox:~/Downloads$ ./withotFork
5000
it took 7 milli seconds
-12 : 1 -->
-11 : 0 -->
-10 : 13 -->
-9 : 0 -->
-8 : 71 --> *
-7 : 0 -->
-6 : 240 --> ****
-5 : 0 -->
-4 : 540 --> *****
-3 : 0 -->
-2 : 913 --> *****
-1 : 0 -->
0 : 1118 --> *****
1 : 0 -->
2 : 1031 --> *****
3 : 0 -->
4 : 635 --> *****
5 : 0 -->
6 : 315 --> *****
7 : 0 -->
8 : 103 --> **
9 : 0 -->
10 : 18 -->
11 : 0 -->
12 : 2 -->
```

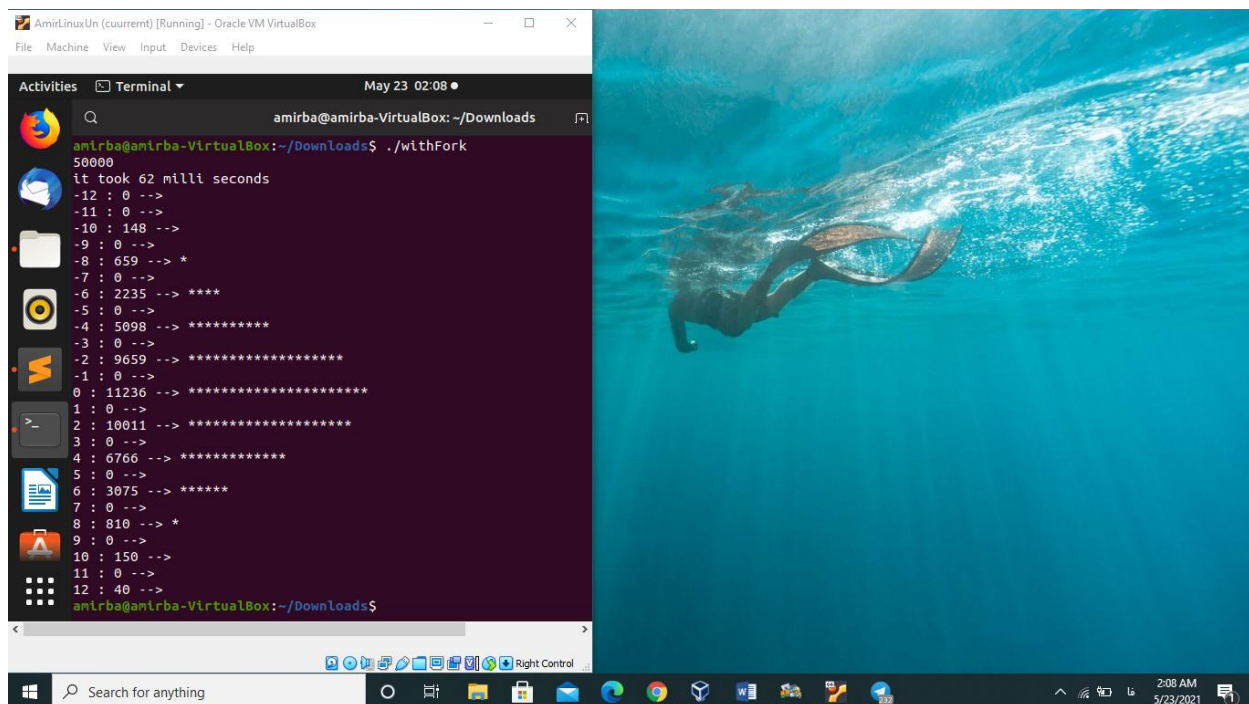


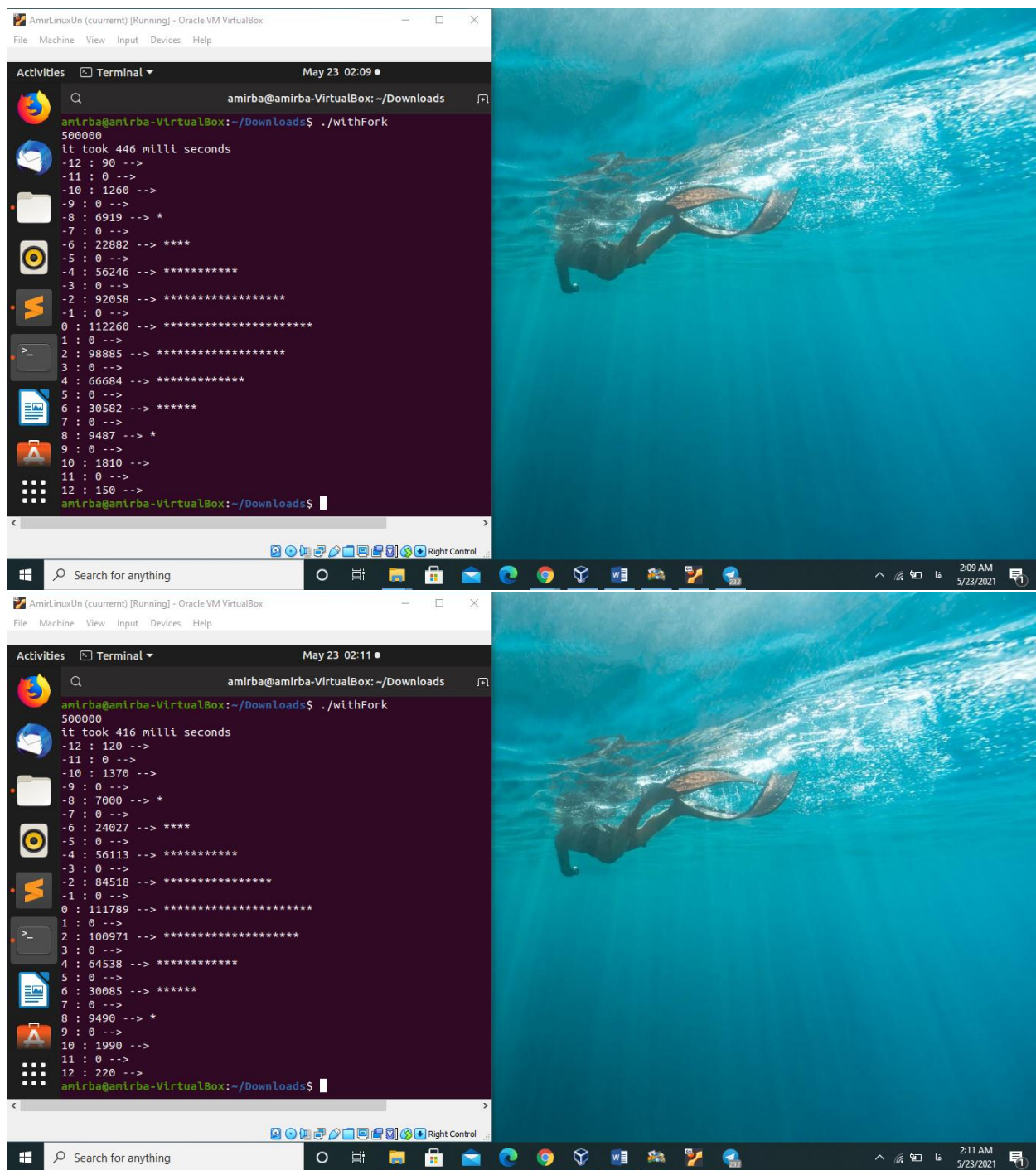
با فورک :

7=5000 میلی

62=50000

416=500000





با توجه به بالا در 5000 تقریباً سرعت ها برابر اند در 50000 افزایش 20 میلی ثانیه مشاهده میشود و در 500000 زمان 3000 میلی ثانیه کاهش مییابد و در مقادیر بالاتر این روند بیتر میشود