دستور کار آزمایشگاه سیستم های عامل

آزمایشگاه سیستم های عامل

آرین محسنی

آبان 1403

دستور کار آزمایشگاه سیستم های عامل

تعریف مسئله:

در این آزمایش هدف آن است که با استفاده از نمونه برداری، نمودار توزیع نرمال ترسیم شود. در ابتدا یک آرایه با نام hist که ۲۵ خانه دارد بسازید. از این آرایه برای نگهداری نتایج آزمایش استفاده می‌شود. این ۲۵ خانه نمایندگان اعداد -۱۲ و +۱۲ هستند. فرآیند نمونه برداری به این صورت است که مقدار ابتدایی متغیر counter شما با مقدار صفر شروع می‌شود و شما بایستی در ۱۲ مرحله و در هر مرحله یک عدد تصادفی بین ۰ تا ۱۰۰ تولید کنید. اگر این عدد تصادفی بزرگتر یا مساوی ۴۹ بود، مقدار counter را یکی افزایش دهید و برعکس. پس از پایان ۱۲ مرحله، بر اساس مقدار counter خانه مربوطه از آرایه hist را افزایش دهید.

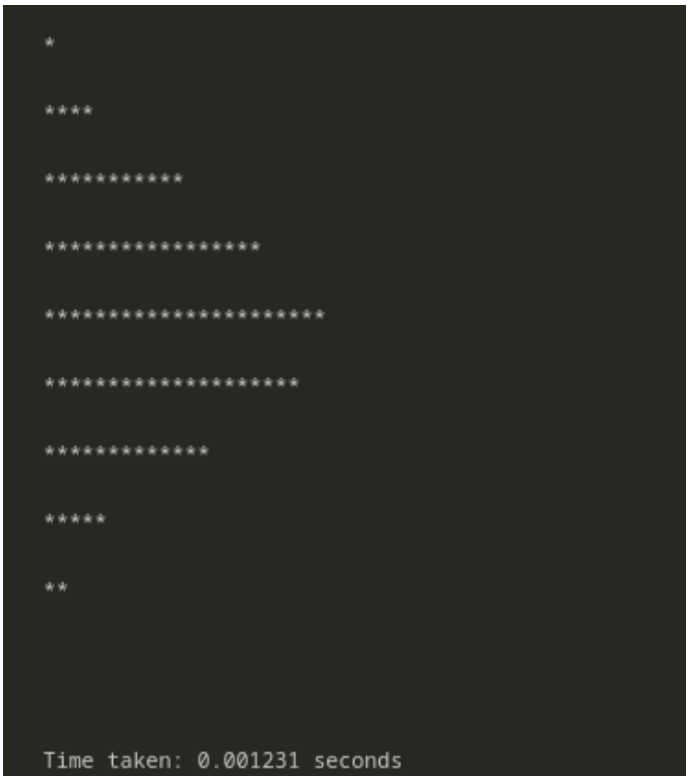
در ابتدا کدی در راستای این مسئله آماده شده است.

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

#define NUM_ITERATIONS 5000000

void printHistogram(int *hist)
{
    int i, j;
    for (i = 0; i < 25; i++)
    {
        for (j = 0; j < hist[i] * 100 / NUM_ITERATIONS; j++)
        {
            printf("*");
        }
        printf("\n");
    }
}

int main()
{
    int hist[25] = {0};

    srand(time(NULL));
    int counter = 0;

    // Start time measurement
    struct timeval start, end;
    gettimeofday(&start, NULL);

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

    // End time measurement
    gettimeofday(&end, NULL);
    double time_taken = (end.tv_sec - start.tv_sec) + (end.tv_usec - start.tv_usec) / 1000000.0;

    printHistogram(hist);
    printf("Time taken: %f seconds\n", time_taken);

    return 0;
}
```

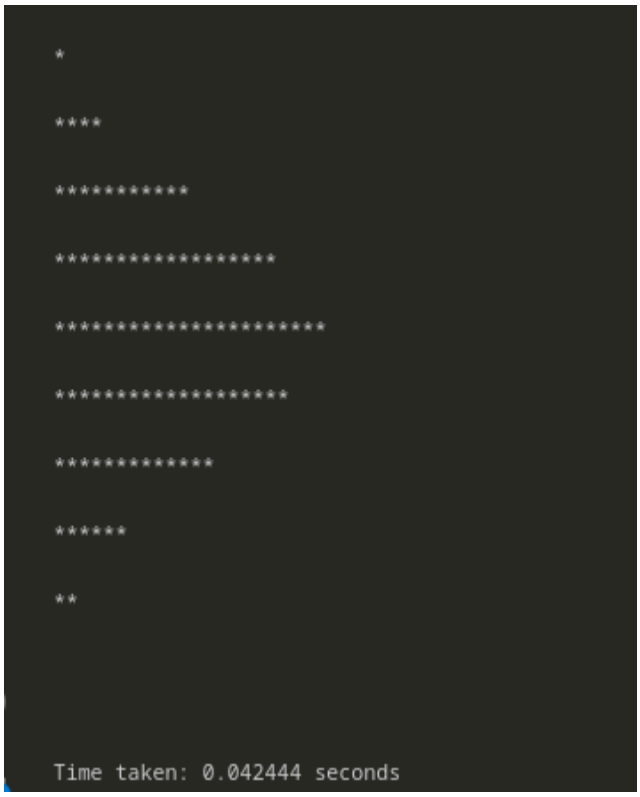
دستور کار آزمایشگاه سیستم های عامل

در این کد با تعداد نمونه های مختلف چنین زمان هایی دریافت میکنیم

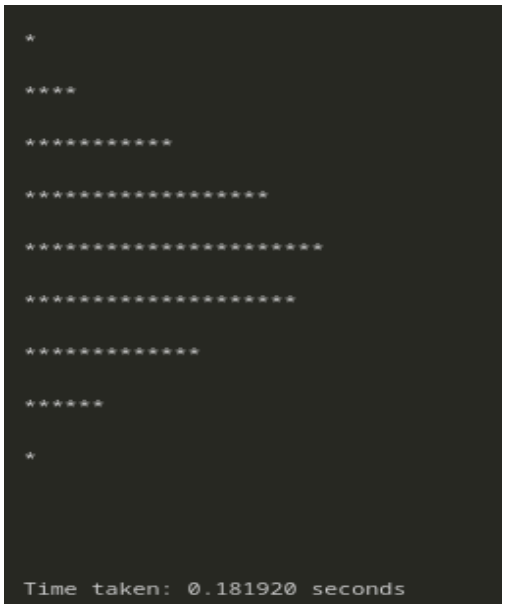
5000



500000



500000



دستور کار آزمایشگاه سیستم های عامل

۲. حال برنامه ای بنویسید که با استفاده از fork() و یا exec() تعدادی فرآیند فرزند ایجاد شود و کارها را پخش کنید. قطعه کد زیر مثالی از نحوه استفاده از fork() است. خروجی این کد در زیر آن آمده است.

می خواهیم با استفاده از fork و exec فرآیند محاسبه توزیع نرمال خود را سریع تر کنیم.

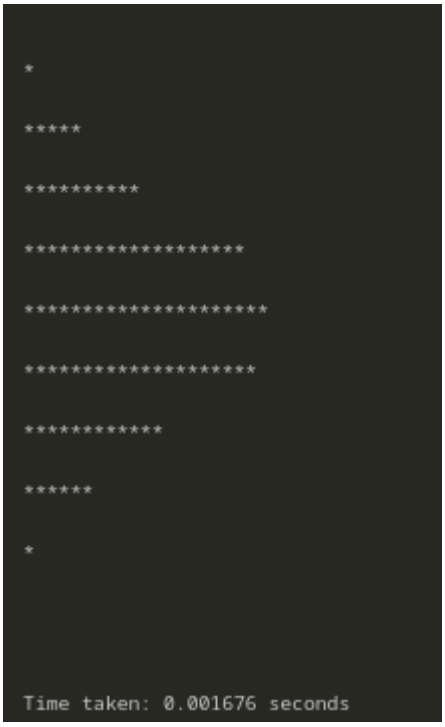
```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <sys/ipc.h>
#include <sys/shm.h>
#include <sys/wait.h>
#include <unistd.h>
#include <sys/time.h>

#define NUM_CHILDREN 2
#define NUM_ITERATIONS 5e5

void printHistogram(int *hist)
{
    int i, j;
    for (i = 0; i < 25; i++)
    {
        for (j = 0; j < hist[i] * 100 / NUM_ITERATIONS; j++)
        {
            printf("*");
        }
        printf("\n");
    }
}

int main() {
    int shmid;
    int *hist;
    int n_per_child = NUM_ITERATIONS / NUM_CHILDREN;
    int counter;
    pid_t pid;

    // Create a shared memory segment
    shmid = shmget(IPC_PRIVATE, 25 * sizeof(int), IPC_CREAT | 0666);

    hist = (int *)shmat(shmid, NULL, 0);

```

دستور کار آزمایشگاه سیستم های عامل

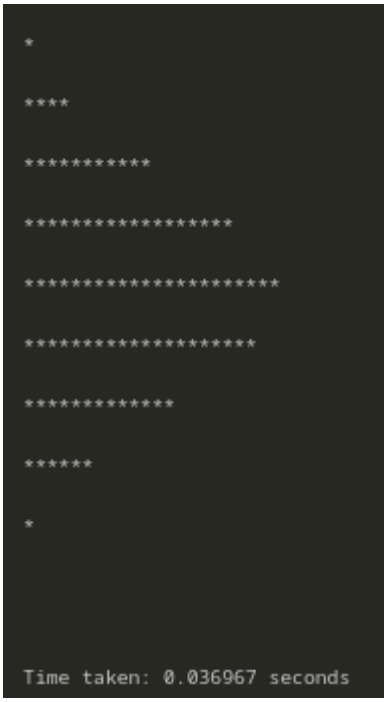
```
38      for (int i = 0; i < 25; i++) {
39          hist[i] = 0;
40      }
41
42      srand(time(NULL));
43
44      // Start time measurement
45      struct timeval start, end;
46      gettimeofday(&start, NULL);
47
48      for (int child = 0; child < NUM_CHILDREN; child++) {
49          pid = fork();
50
51          // printf("Pid :%d\n",getpid());
52
53          if (pid == 0) { // Child process
54
55              printf("Pid :%d\n",getpid());
56              for (int iter = 0; iter < n_per_child; iter++) {
57                  counter = 0;
58                  for (int i = 0; i < 12; i++) {
59                      int random_number = rand() % 100;
60                      if (random_number >= 49)
61                          counter++;
62                      else
63                          counter--;
64                  }
65                  // Avoid race conditions
66                  __sync_fetch_and_add(&hist[counter + 12], 1);
67              }
68              shmdt(hist);
69              exit(0);
70          }
71      }
72
73      // Wait for all children
74      for (int i = 0; i < NUM_CHILDREN; i++) {
75          wait(NULL);
76      }
77
78      // End time measurement
79      gettimeofday(&end, NULL);
80      double time_taken = (end.tv_sec - start.tv_sec) + (end.tv_usec - start.tv_usec) / 1000000.0;
81
82      printHistogram(hist);
83      printf("Time taken: %f seconds\n", time_taken);
84
85      shmdt(hist);
```

دستور کار آزمایشگاه سیستم های عامل

5000



50000



500000

مشاهده شد که سریع تر از حالت قبل محاسبه شد.