به نام ایزد یکتا

آزمایش پنجم درس آزمایشگاه سیستم عامل

­­

استاد: مهندس قاسمی

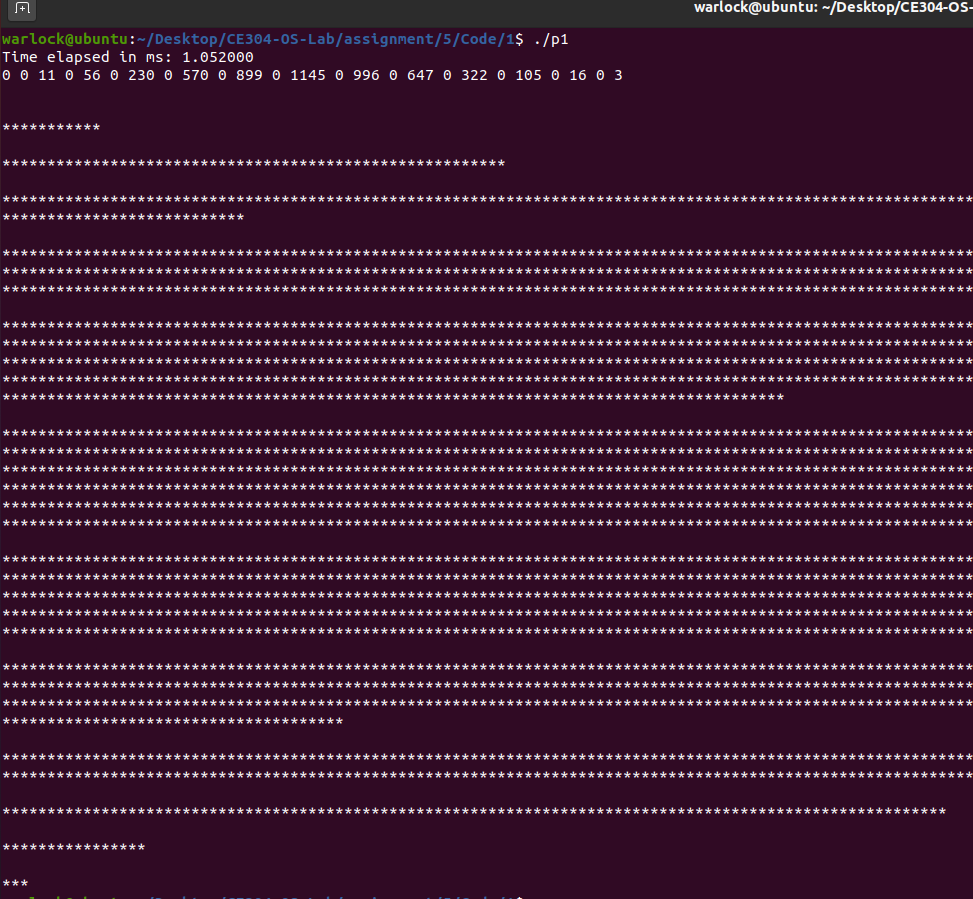
تهیه کننده: بردیا اردکانیان

۹۸۳۱۰۷۲

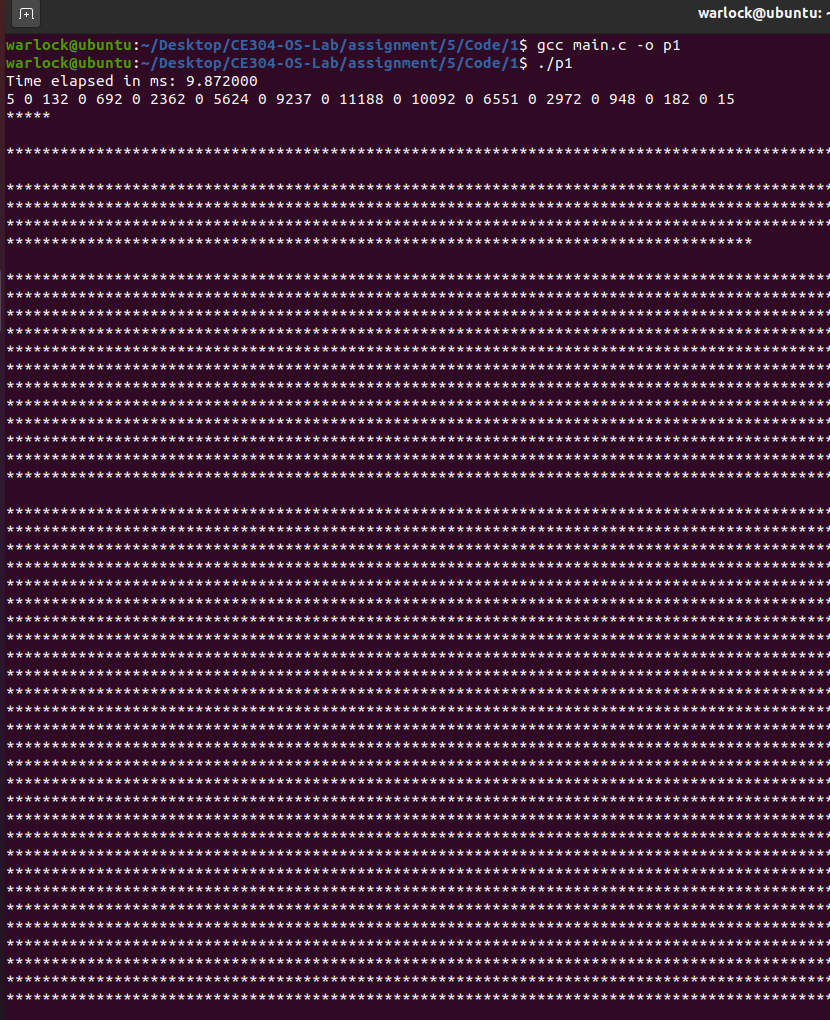
**آزمایش اول)**

در این بخش طبق توضیحات دستور کار، یک آرایه به نام hist ایجاد می‌کنیم که ۲۵ خانه دارد و ابتدا تمامی خانه‌ها را به صفر مقدار دهی می‌کنیم. حال آزمایش را به تعداد نمونه (که در جدول آمده است) تکرار می‌کنیم. این آزمایش چیست؟ این است که در هر بار کانتر را صفر کنیم و ۱۲ بار یک عدد رندوم بین ۰ تا ۱۰۰ تولید کنیم، اگر بزرگتر مساوی ۴۹ باشد، کانتر را یکی زیاد می‌کنیم و اگر کمتر باشد، کانتر را یکی کم می‌کنیم. حال در آخر کانتر عددی بین -۱۲ و ۱۲ است و می‌خواهیم شماره خانه ی کانتر از آرایه hist را یکی زیاد کنیم. برای اینکار چون ایندکس منفی نداریم، خانه ی کانتر + ۱۲ را یکی زیاد می‌کنیم.در آخر آرایه hist تشکیل شده است و با استفاده از تکه کد داخل دستور کار، نمودار توزیع نرمال را رسم می کنیم. همینطور برای به دست آوردن زمان اجرا از clock استفاده می‌کنیم و در ابتدا و انتهای برنامه آن را می‌گذاریم.

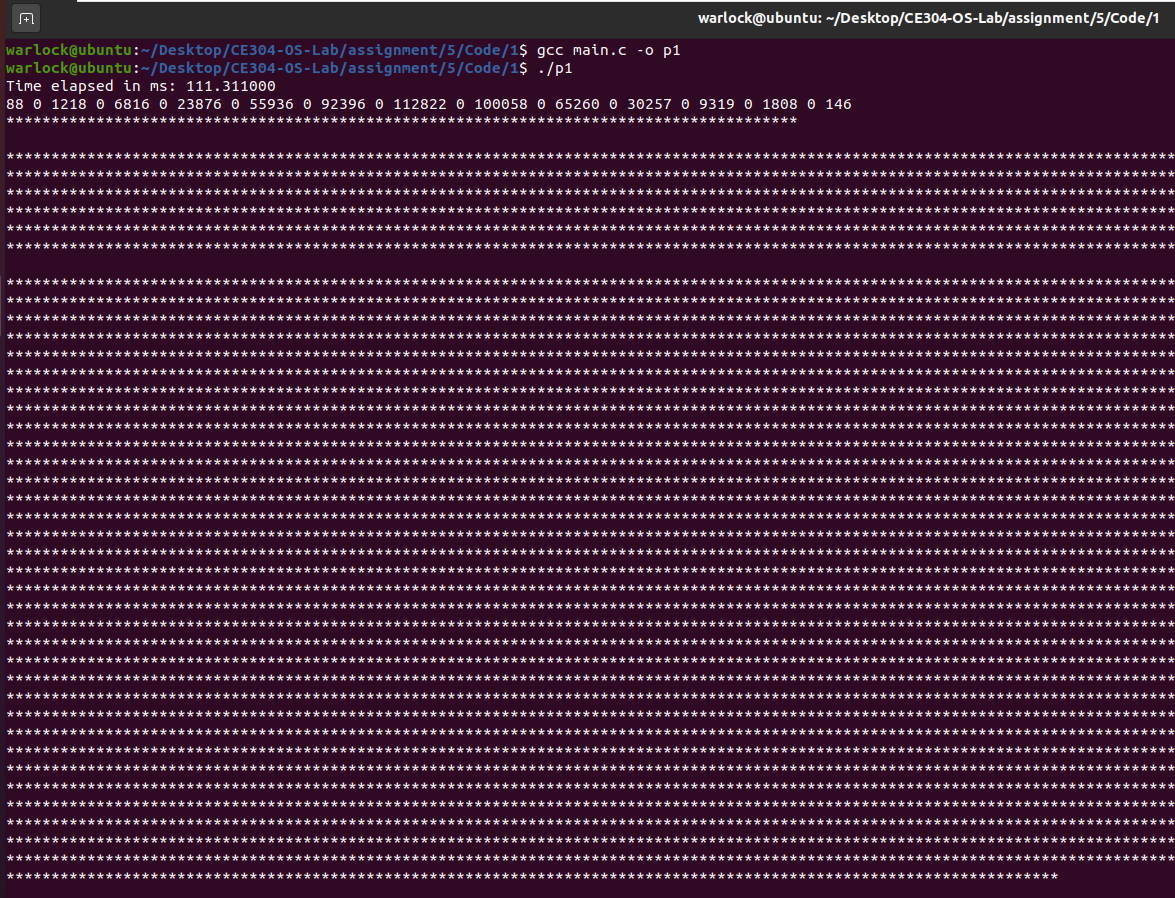
|  |  |  |  |
| --- | --- | --- | --- |
| **تعداد نمونه** | **5000** | **50000** | **500000** |
| **زمان اجرا** | **1.052000** | **9.872000** | **111.311000** |

****

**عکس 1-1 (5000 ورودی)**

****

**عکس 1-2 (50000 ورودی)**

****

**عکس 3-1 (500000 ورودی)**

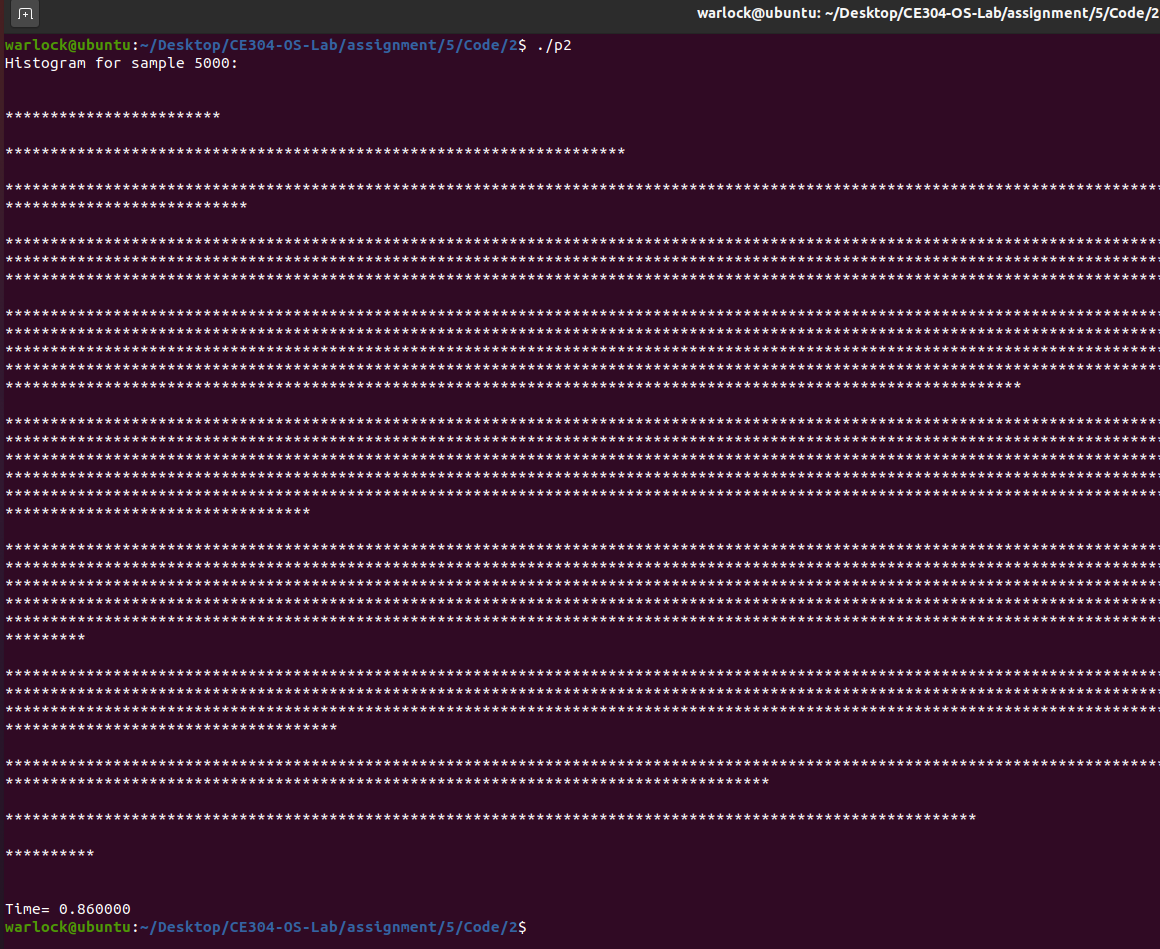
کد بخش اول:

*#include <stdio.h>  
#include <stdlib.h>  
#include <time.h>  
  
int* ***main****()  
{  
    clock\_t start = clock();  
    int hist[25];  
    for (int i = 0; i < 25; i++)  
    {  
        hist[i] = 0;  
    }  
    int counter = 0;  
    srand(time(NULL));  
    for (int i = 0; i < 5000; i++)  
    {  
        counter = 0;  
        for (int j = 0; j < 12; j++)  
        {  
            int random\_number = rand() % 100;  
            if (random\_number >= 49)  
            {  
                counter++;  
            }  
            else  
            {  
                counter--;  
            }  
        }  
        hist[counter + 12]++;  
    }  
    clock\_t stop = clock();  
    double elapsed = (double)(stop - start) \* 1000.0 / CLOCKS\_PER\_SEC;  
    printf("Time elapsed in ms: %f\n", elapsed);  
    for (int i = 0; i < 25; i++)  
    {  
        printf("%d ", hist[i]);  
    }  
    printf("\n");  
    int i, j;  
    for (i = 0; i < 25; i++)  
    {  
        for (j = 0; j < hist[i]; j++)  
        {  
            printf("\*");  
        }  
        printf("\n");  
    }  
    return 0;  
}*

**آزمایش دوم)**

در این بخش با استفاده از یک فورک از پردازه فرزند کمک می‌گیریم تا نیمی از محاسبات را او انجام دهد و در یک حافظه مشترک هم پدر و هم فرزند این مقادیر را قرار دهند و در نهایت زمان را محاسبه می کنیم و نمودار آن را مثل قسمت قبلی رسم می کنیم. با تقسیم کارها بین پردازه پدر و فرزند باعث می شود زمان کمتری طول بکشد و مشاهده می شود که هر چه تعداد نمونه بیشتر می شود این تاثیر نیز بیشتر می شود. زمان های به دست آمده به شرح زیر است:

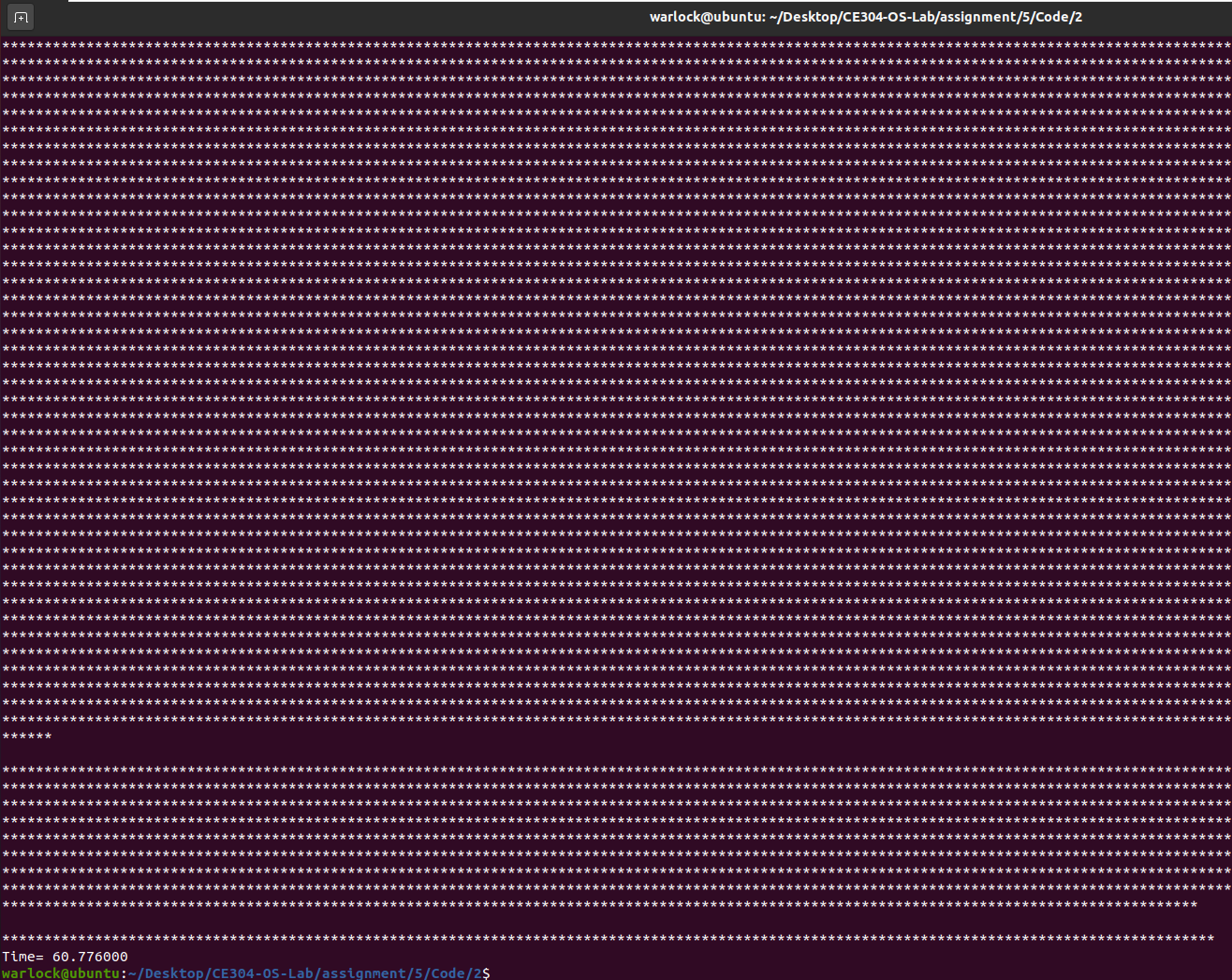
|  |  |  |  |
| --- | --- | --- | --- |
| **تعداد نمونه** | **5000** | **50000** | **500000** |
| **زمان اجرا** | **0.860000** | **6.871000** | **60.776000** |
| **درصد افزایش سرعت** | **18.25** | **30.39** | **45.39** |

****

**عکس 2-1**

****

**عکس 2-2**

****

**عکس 2-3**

**کد بخش دوم:**

#include <stdio.h>  
#include <time.h>  
#include <stdlib.h>  
#include <unistd.h>  
#include <sys/types.h>  
#include <sys/ipc.h>  
#include <sys/shm.h>  
#include <sys/stat.h>  
#include <semaphore.h>  
#include <pthread.h>  
  
#define SHM\_KEY 102030  
#define SHM\_SIZE sizeof(int) \* 25  
  
sem\_t mutex;  
  
int **create\_segment**()  
{  
    int shmid;  
    if ((shmid = shmget(IPC\_PRIVATE, SHM\_SIZE, S\_IRUSR | S\_IWUSR)) == -1)  
    {  
        printf("error error error!");  
        perror("shmget");  
        exit(1);  
    }  
    return shmid;  
}  
  
int \***attach\_segment**(int shmid)  
{  
    int \*shm;  
    if ((shm = shmat(shmid, NULL, 0)) == (int \*)-1)  
    {  
        perror("Adding address space failed");  
        exit(EXIT\_FAILURE);  
    }  
    return shm;  
}  
  
void **detach\_segment**(int \*shm)  
{  
    shmdt(shm);  
}  
  
void **remove\_segment**(int shmid)  
{  
    if (shmctl(shmid, IPC\_RMID, NULL) == -1)  
    {  
        perror("Removing failed");  
        exit(EXIT\_FAILURE);  
    }  
}  
  
void **print\_histogram**(int \*hist, int number\_of\_samples)  
{  
    printf("Histogram for sample %d:\n", number\_of\_samples);  
    for (int i = 0; i < 25; i++)  
    {  
        for (int j = 0; j < hist[i]; j++)  
        {  
            printf("\*");  
        }  
        printf("\n");  
    }  
}  
  
double **calculate**(int number\_of\_samples)  
{  
    clock\_t begin = clock();  
    int shmid = create\_segment();  
    int \*hist = attach\_segment(shmid);  
    srand(time(0));  
    int rand\_num, counter;  
    if (fork() == 0)  
    {  
        for (int i = 0; i < number\_of\_samples / 2; i++)  
        {  
            counter = 0;  
            for (int j = 0; j < 12; j++)  
            {  
                rand\_num = rand() % 100;  
                if (rand\_num >= 49)  
                    counter += 1;  
                else  
                    counter -= 1;  
            }  
            hist[counter + 12] += 1;  
        }  
        exit(0);  
    }  
    else  
        for (int i = 0; i < number\_of\_samples / 2; i++)  
        {  
            counter = 0;  
            for (int j = 0; j < 12; j++)  
            {  
                rand\_num = rand() % 100;  
                if (rand\_num >= 49)  
                    counter += 1;  
                else  
                    counter -= 1;  
            }  
            hist[counter + 12] += 1;  
        }  
    print\_histogram(hist, number\_of\_samples);  
    detach\_segment(hist);  
    remove\_segment(shmid);  
    clock\_t end = clock();  
    double time\_spend = (double)(end - begin) \* 1000.0 / CLOCKS\_PER\_SEC;  
    return time\_spend;  
}  
  
int **main**()  
{  
    double time\_spend = calculate(500000);  
    printf("Time= %f\n", time\_spend);  
    return 0;  
}