

به نام خدا

گزارش آزمایش ۵

آزمایشگاه سیستم عامل

استاد : حسینی تودشکی

سید سعید شفیعی

آبان ۱۴۰۰

بخش اول :

قصد داریم که نمونه برداری مساله را از روی تکه کدی که به صورت سریال نوشته شده است، انجام بدهیم. عناصر ۰ تا ۱۱ آرایه hist را نمایندگان اعداد -۱۲ تا ۱ - در نظر گرفته ایم و عناصر ۱۲ تا ۲۴ نیز، نمایندگان اعداد ۰ تا ۱۲ هستند به همین دلیل زمانی که با استفاده از counter می‌خواهیم به یک خانه از آرایه ی hist که نماینده مقدار counter است مقدار دهی کنیم، کافی است که به خانه $\text{counter} + 12$ دسترسی پیدا کنیم.

برای نمایش زمان سپری شده برای پردازش برنامه می توان از تابع $\text{clock}()$ بهره برد.

به ازای ورودی های داده شده درون دستورکار، برنامه را پردازش می کنیم و نتیجه به صورت زیر است :

تعداد نمونه ۵۰۰۰:

```

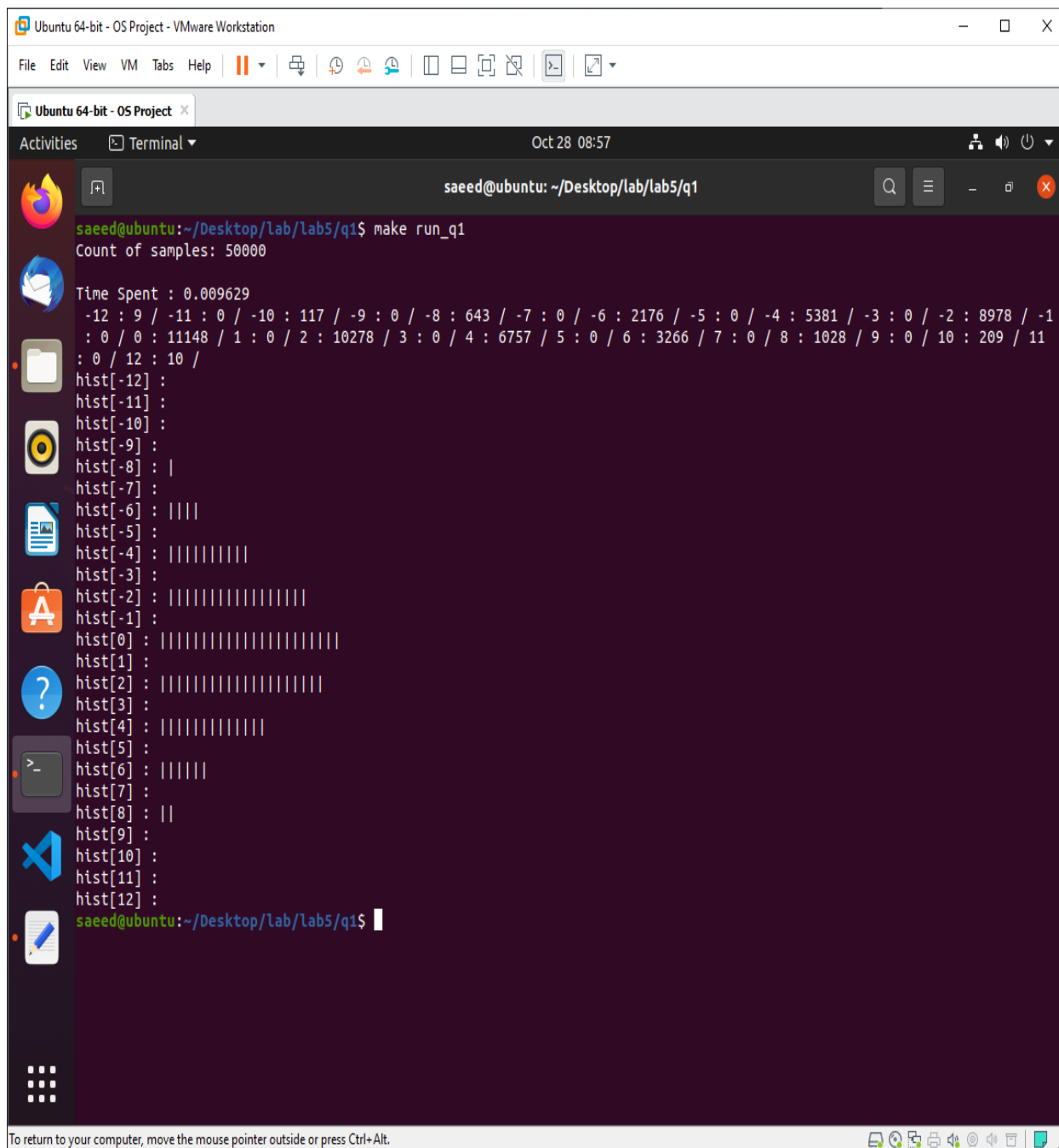
saeed@ubuntu: ~/Desktop/lab/lab5/q1
saeed@ubuntu:~/Desktop/lab/lab5/q1$ make run_q1
Count of samples: 5000

Time Spent : 0.001021
-12 : 2 / -11 : 0 / -10 : 10 / -9 : 0 / -8 : 65 / -7 : 0 / -6 : 238 / -5 : 0 / -4 : 496 / -3 : 0 / -2 : 904 / -1 : 0
/ 0 : 1158 / 1 : 0 / 2 : 1012 / 3 : 0 / 4 : 683 / 5 : 0 / 6 : 323 / 7 : 0 / 8 : 89 / 9 : 0 / 10 : 17 / 11 : 0 / 12 : 3
/
hist[-12] :
hist[-11] :
hist[-10] :
hist[-9] :
hist[-8] : |
hist[-7] :
hist[-6] : ||||
hist[-5] :
hist[-4] : |||||
hist[-3] : |||||
hist[-2] : |||||
hist[-1] : |||||
hist[0] : |||||
hist[1] :
hist[2] : |||||
hist[3] :
hist[4] : |||||
hist[5] :
hist[6] : |||||
hist[7] :
hist[8] : |
hist[9] :
hist[10] :
hist[11] :
hist[12] :
saeed@ubuntu:~/Desktop/lab/lab5/q1$

```

میتوان با جمع کردن نمونه های hist از مقدار iteration های مطمئن شد (کامنت شده درون کد)

تعداد نمونه ۵۰۰۰۰:



```

saeed@ubuntu: ~/Desktop/lab/lab5/q1
saeed@ubuntu:~/Desktop/lab/lab5/q1$ make run_q1
Count of samples: 50000

Time Spent : 0.009629
-12 : 9 / -11 : 0 / -10 : 117 / -9 : 0 / -8 : 643 / -7 : 0 / -6 : 2176 / -5 : 0 / -4 : 5381 / -3 : 0 / -2 : 8978 / -1
: 0 / 0 : 11148 / 1 : 0 / 2 : 10278 / 3 : 0 / 4 : 6757 / 5 : 0 / 6 : 3266 / 7 : 0 / 8 : 1028 / 9 : 0 / 10 : 209 / 11
: 0 / 12 : 10 /
hist[-12] :
hist[-11] :
hist[-10] :
hist[-9] :
hist[-8] : |
hist[-7] :
hist[-6] : ||||
hist[-5] :
hist[-4] : |||||
hist[-3] :
hist[-2] : |||||
hist[-1] :
hist[0] : |||||
hist[1] :
hist[2] : |||||
hist[3] :
hist[4] : |||||
hist[5] :
hist[6] : |||||
hist[7] :
hist[8] : ||
hist[9] :
hist[10] :
hist[11] :
hist[12] :
saeed@ubuntu:~/Desktop/lab/lab5/q1$

```

تعداد نمونه ۵۰۰۰۰۰:

```

saeed@ubuntu: ~/Desktop/lab/lab5/q1
saeed@ubuntu:~/Desktop/lab/lab5/q1$ make run_q1
Count of samples: 500000

Time Spent : 0.092628
-12 : 81 / -11 : 0 / -10 : 1023 / -9 : 0 / -8 : 6229 / -7 : 0 / -6 : 22304 / -5 : 0 / -4 : 53370 / -3 : 0 / -2 : 9052
2 / -1 : 0 / 0 : 112011 / 1 : 0 / 2 : 102169 / 3 : 0 / 4 : 68067 / 5 : 0 / 6 : 31998 / 7 : 0 / 8 : 10057 / 9 : 0 / 10
: 2000 / 11 : 0 / 12 : 169 /
hist[-12] :
hist[-11] :
hist[-10] :
hist[-9] :
hist[-8] : |
hist[-7] :
hist[-6] : ||||
hist[-5] :
hist[-4] : |||||
hist[-3] : |||||
hist[-2] : |||||
hist[-1] : |||||
hist[0] : |||||
hist[1] : |||||
hist[2] : |||||
hist[3] : |||||
hist[4] : |||||
hist[5] : |||||
hist[6] : |||||
hist[7] : |||||
hist[8] : ||
hist[9] :
hist[10] :
hist[11] :
hist[12] :
saeed@ubuntu:~/Desktop/lab/lab5/q1$

```

با توجه به هر ۳ مورد به دست آمده بیشترین فراوانی متعلق به عدد ۰ است. این به این معنا است که در بیشتر iteration ها، مقدار counter برابر با ۰ بوده است.

تعداد نمونه	5000	50000	500000
زمان اجرا(ثانیه)	0.001021	0.009629	0.092628

بخش دوم:

در بخش دوم با استفاده از `fork()` تعدادی فرزند ایجاد می کنیم و پردازش ها را میان آنها پخش می کنیم. از مفهوم `shared-memory` هم برای اینکه همه پردازش ها از حافظه اشتراکی استفاده کنند استفاده کرده ایم. کد را `Makefile` کامپایل کرده و به ازای تعداد نمونه های مختلف زمان را به دست می آوریم.

تعداد نمونه ۵۰۰۰:

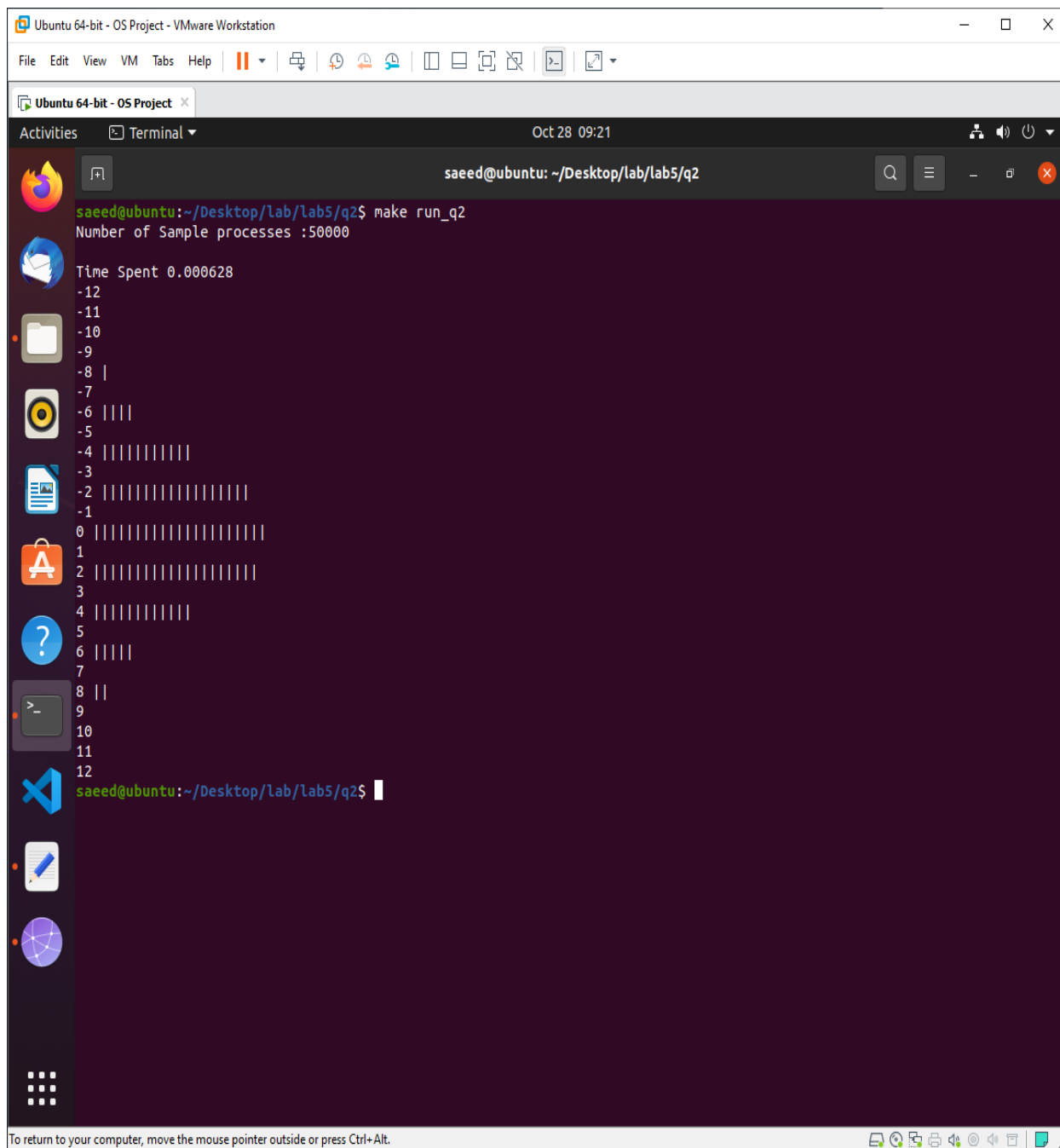
```

saeed@ubuntu: ~/Desktop/lab/lab5/q2
saeed@ubuntu:~/Desktop/lab/lab5/q2$ make run_q2
Number of Sample processes :5000

Time Spent 0.000692
-12
-11
-10
-9
-8 ||
-7
-6 |||||
-5
-4 |||||
-3 |||||
-2 |||||
-1 |||||
0 |||||
1 |||||
2 |||||
3 |||||
4 |||||
5 |||||
6 |||||
7
8 |
9
10
11
12
saeed@ubuntu:~/Desktop/lab/lab5/q2$

```

تعداد نمونه ۵۰۰۰۰:



```
saheed@ubuntu: ~/Desktop/lab/lab5/q2
saheed@ubuntu:~/Desktop/lab/lab5/q2$ make run_q2
Number of Sample processes :50000

Time Spent 0.000628
-12
-11
-10
-9
-8 |
-7
-6 ||||
-5
-4 |||||
-3
-2 |||||
-1
0 |||||
1
2 |||||
3
4 |||||
5
6 ||||
7
8 ||
9
10
11
12
saheed@ubuntu:~/Desktop/lab/lab5/q2$
```

تعداد نمونه ۵۰۰۰۰۰:

```

saeed@ubuntu: ~/Desktop/lab/lab5/q2
saeed@ubuntu:~/Desktop/lab/lab5/q2$ make run_q2
Number of Sample processes :500000

Time Spent 0.000722
-12
-11
-10
-9
-8 |
-7
-6 ||||
-5
-4 |||||
-3 |||||
-2 |||||
-1 |||||
0 |||||
1 |||||
2 |||||
3 |||||
4 |||||
5 |||||
6 |||||
7
8 |
9
10
11
12
saeed@ubuntu:~/Desktop/lab/lab5/q2$

```

50000	50000	5000	تعداد نمونه
0.000722	0.000628	0.000692	زمان اجرا(ثانیه)

بخش سوم:

بله ، زیرا در این برنامه دارای پردازش های مختلفی است که هر کدام می تواند به متغیر های مشترک دسترسی داشته باشند و آن را تغییر دهند. متغیر `hist[counter + 12]` این متغیر است. برای جلوگیری از این اتفاق می توان با استفاده از `spin lock` یا `semaphore` انحصار متقابل در حین دسترسی به آرایه `hist` ایجاد کرد.

`Semaphore` در واقع یک مکانیزم سیگنالی است که به صورت یک متغیر بین ریسمان های برنامه به اشتراک گذاشته می شود. روی این متغیر می توان از دو تابع `wait` و `signal` استفاده کرد. با صدا زدن تابع `wait` مقدار آن ۱ واحد اضافه می شود و با صدا زدن تابع `signal` از مقدار آن ۱ واحد کاسته می شود. اگر مقدار این متغیر صفر شود ، دیگر نمیتوان روی آن تابع `wait` را صدا زد. پس پردازش ای که `wait` را صدا کرده اجرایش متوقف می شود تا زمانی که پردازش دیگر مقدار `signal` را صدا بزند.

بخش چهارم:

واضح است که با مقایسه به دلیل موازی سازی و همروندی و تقسیم کارها در برنامه سرعت بخش دوم افزایش دارد.

تعداد نمونه	5000	50000	500000
افزایش سرعت	83%	87%	91%