به نام خدا

گزارش آزمایش 5

آزمایشگاه سیستم عامل

استاد : حسینی تودشکی

سید سعید شفیعی

آبان 1400

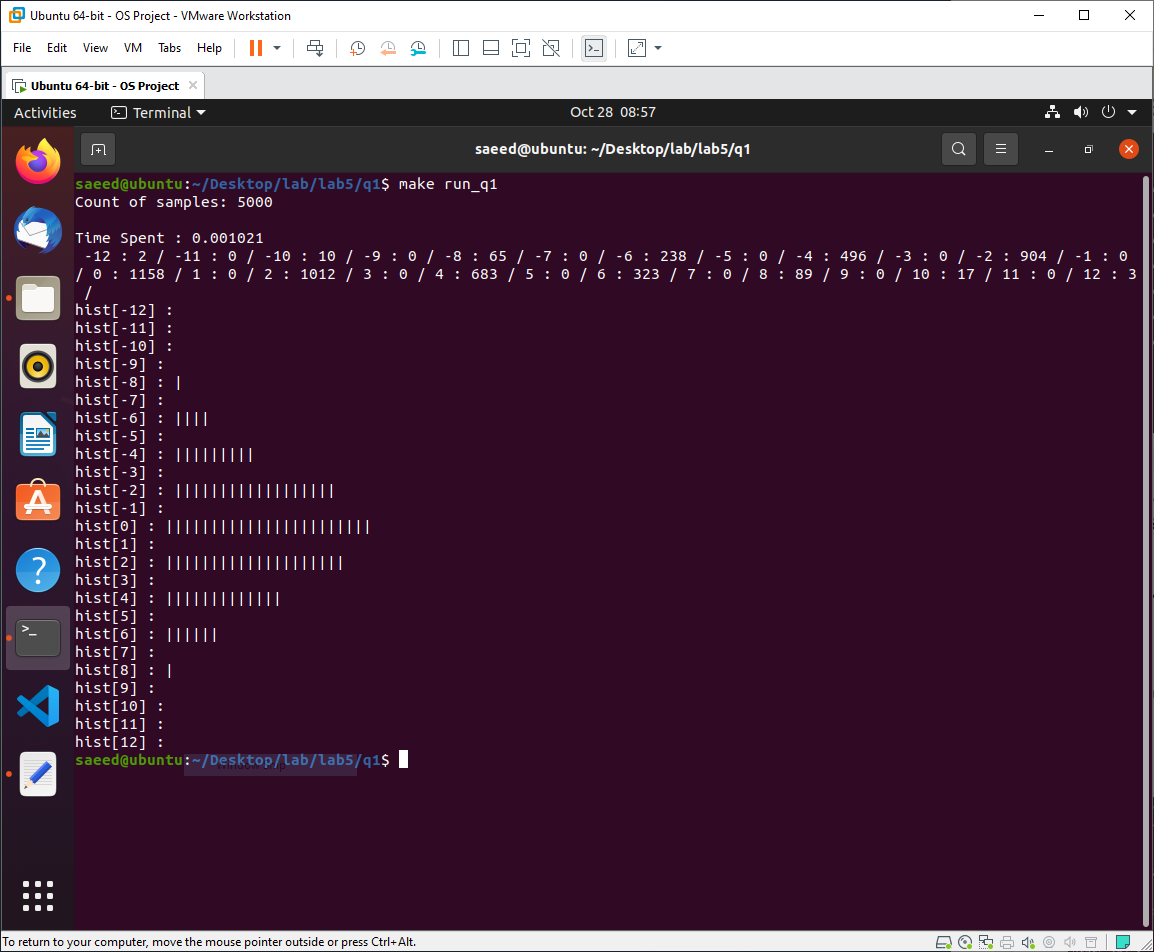
بخش اول :

قصد داریم که نمونه برداری مساله را از روی تکه کدی که به صورت سریال نوشته شده است، انجام بدهیم. عناصر 0 تا 11 آرایه hist را نمایندگان اعداد 12 -تا 1 - در نظر گرفته ایم و عناصر 12 تا 24 نیز، نمایندگان اعداد 0 تا 12 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 counter + 12 دسترسی پیدا کنیم.

برای نمایش زمان سپری شده برای پردازش برنامه می توان از تابع clock() بهره برد.

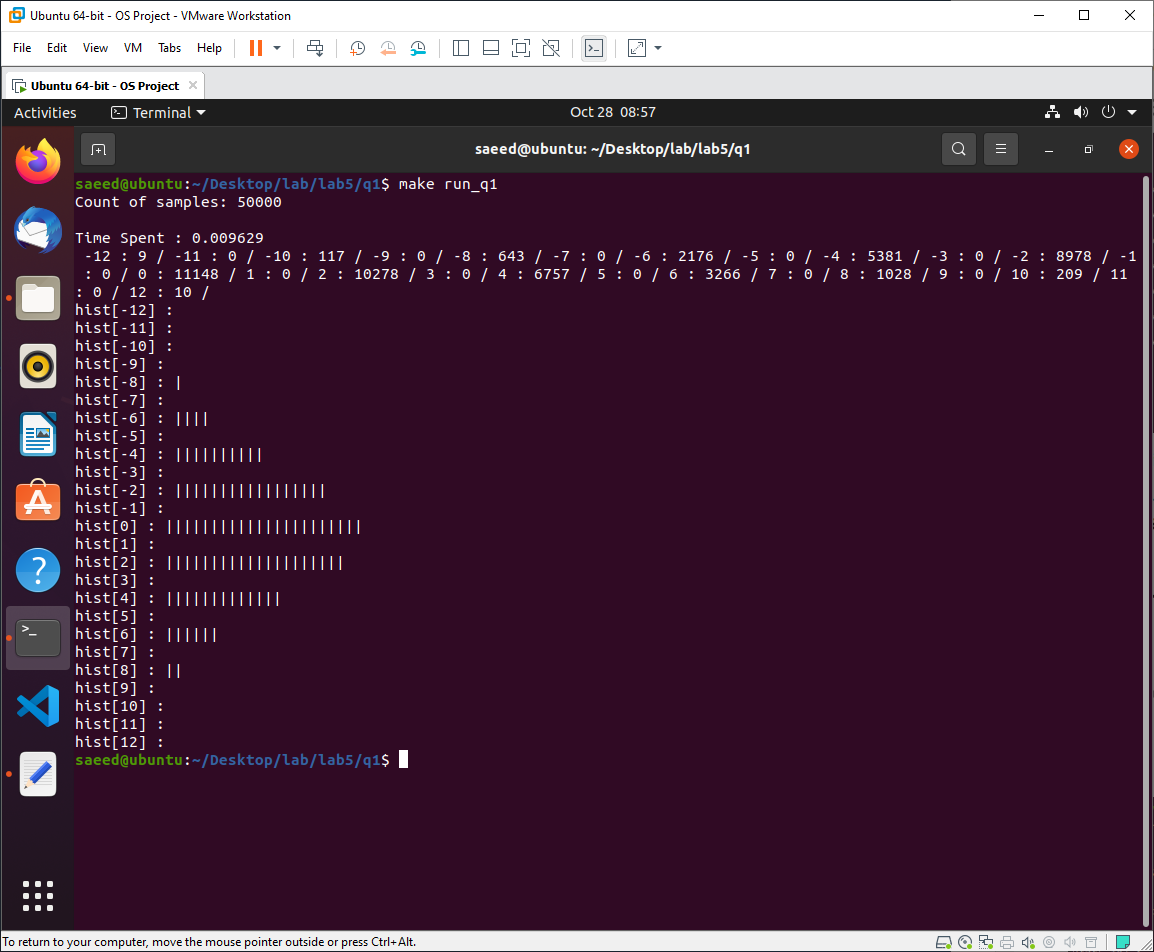
به ازای ورودی های داده شده درون دستورکار، برنامه را پردازش می کنیم و نتیجه به صورت زیر است :

تعداد نمونه 5000:

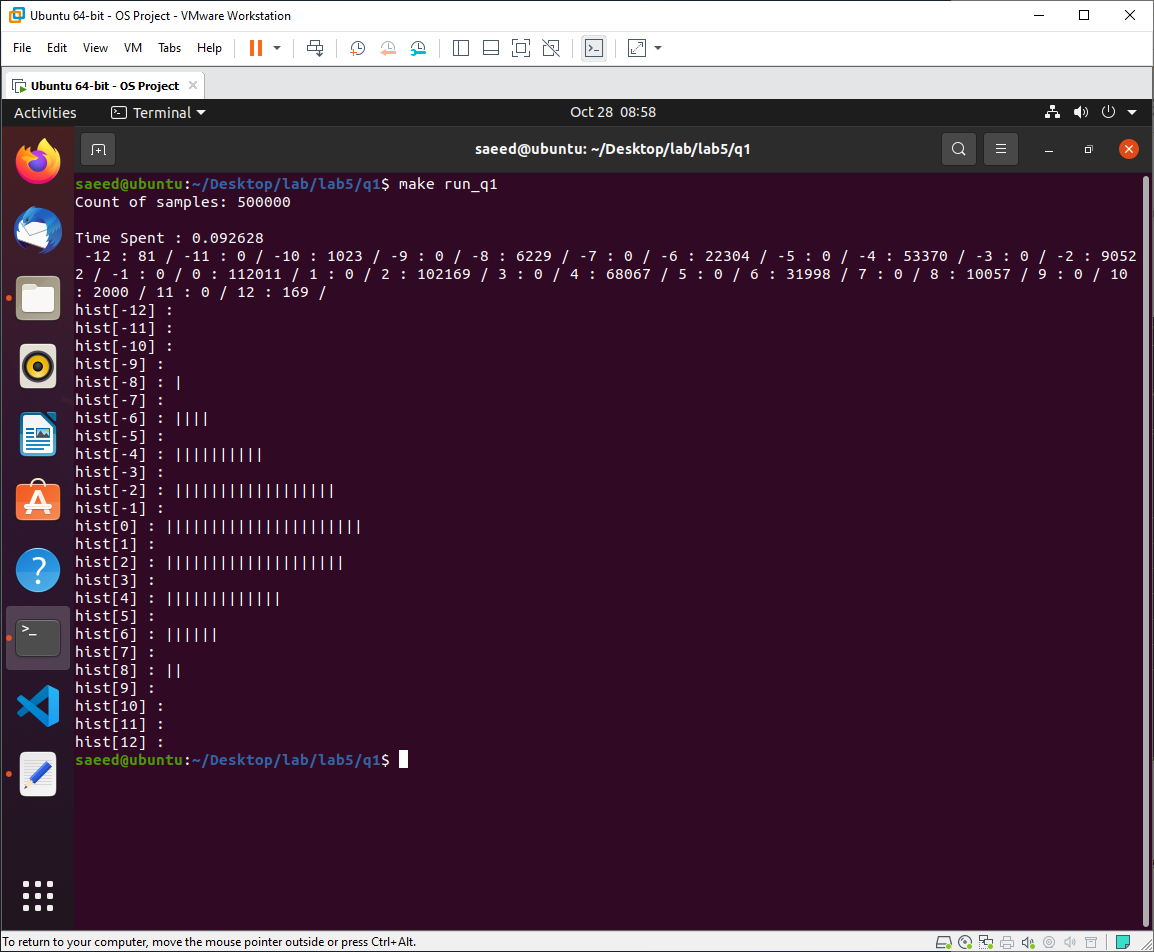


میتوان با جمع کردن نمونه های hist از مقدار iteration های مطمئن شد ( کامنت شده درون کد)

تعداد نمونه 50000:



تعداد نمونه 500000:



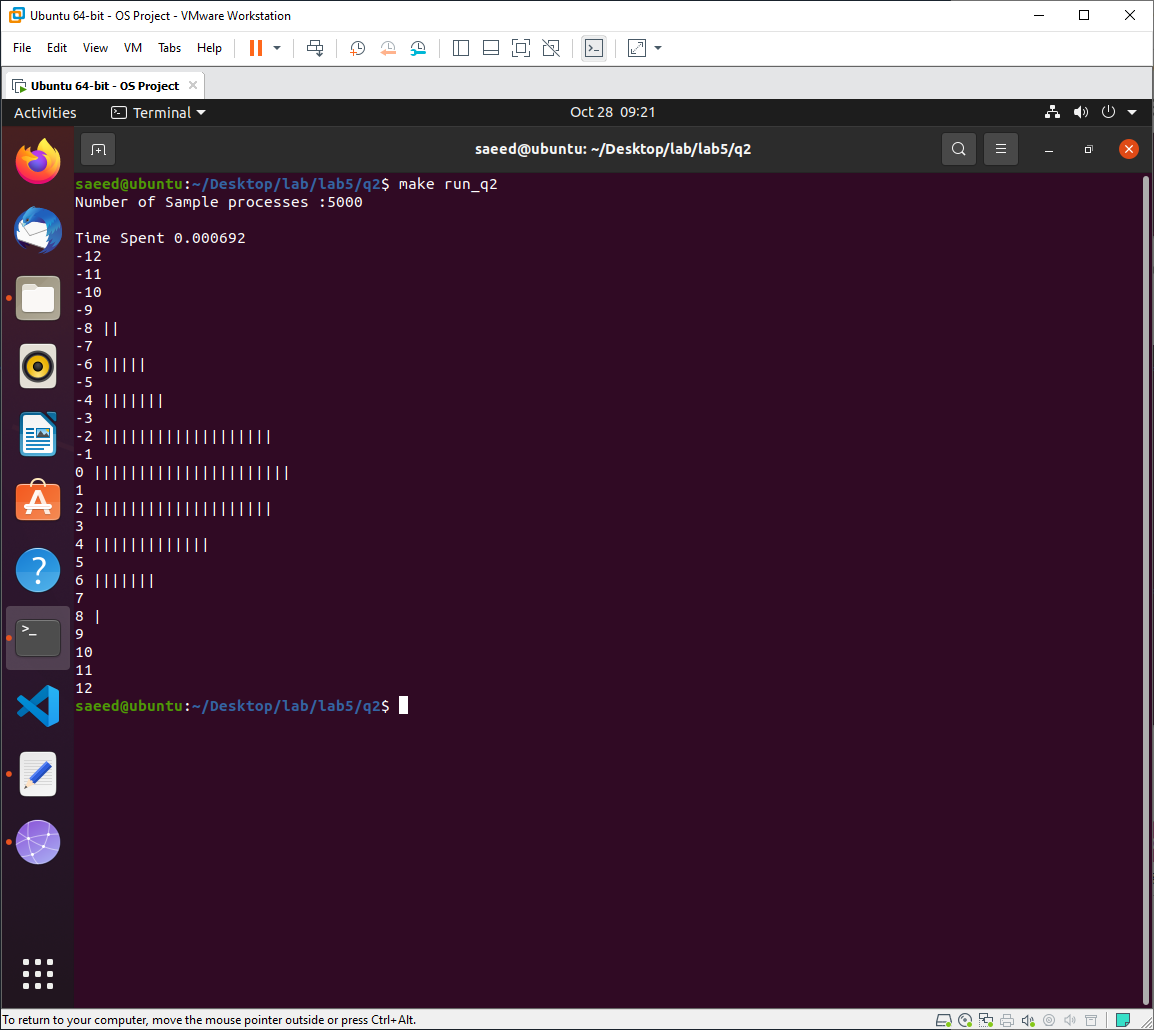
با توجه به هر 3 مورد به دست آمده بیشترین فراوانی متعلق به عدد 0 است. این به این معنا است که در بیشتر iteration ها، مقدار counter برابر با 0 بوده است.

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

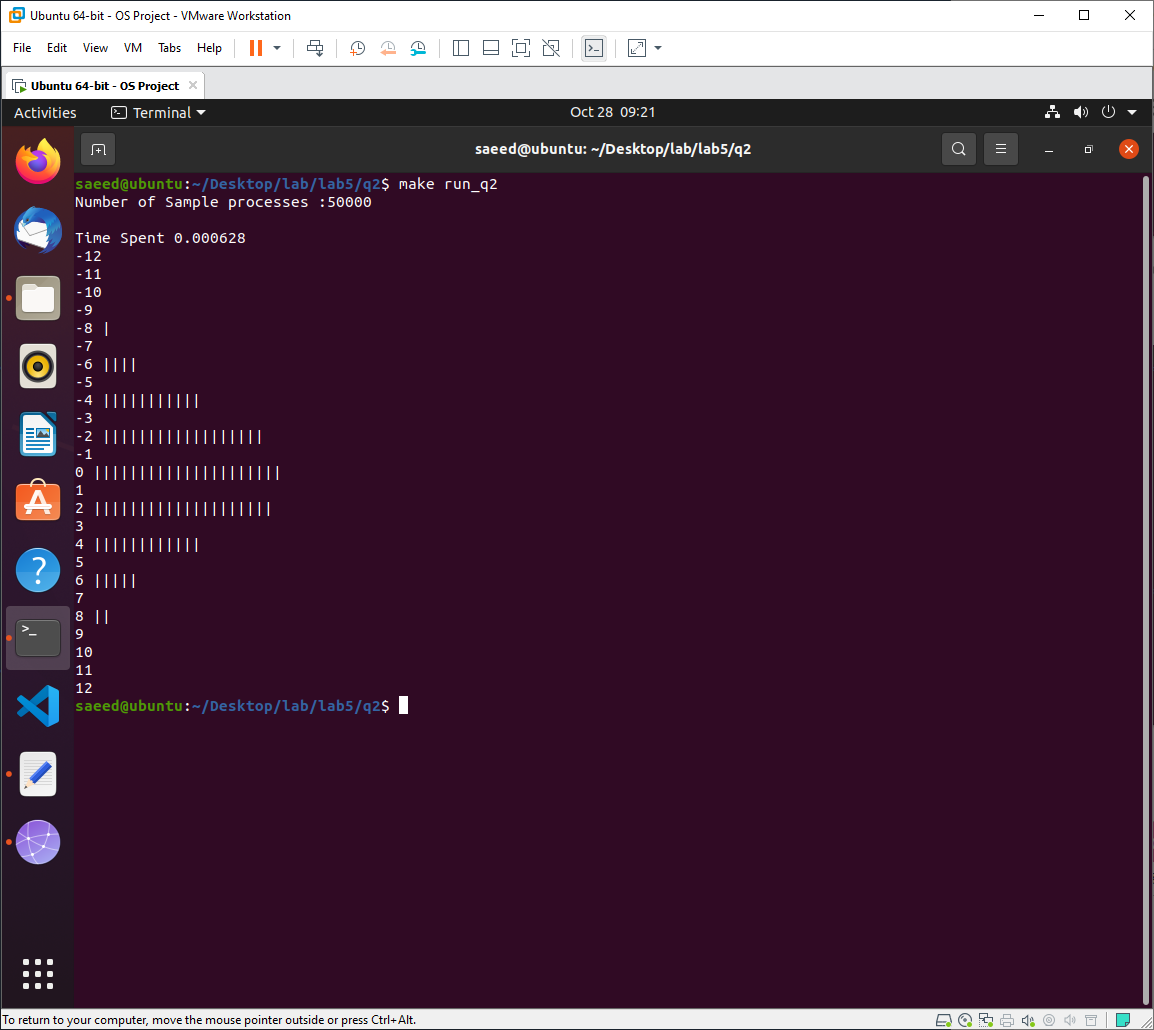
بخش دوم:

در بخش دوم با استفاده از fork() تعدادی فرآیند فرزند ایجاد می کنیم و پردازه ها را میان آنها پخش می کنیم. از مفهوم shared-memory هم برای اینکه همه پردازه 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.

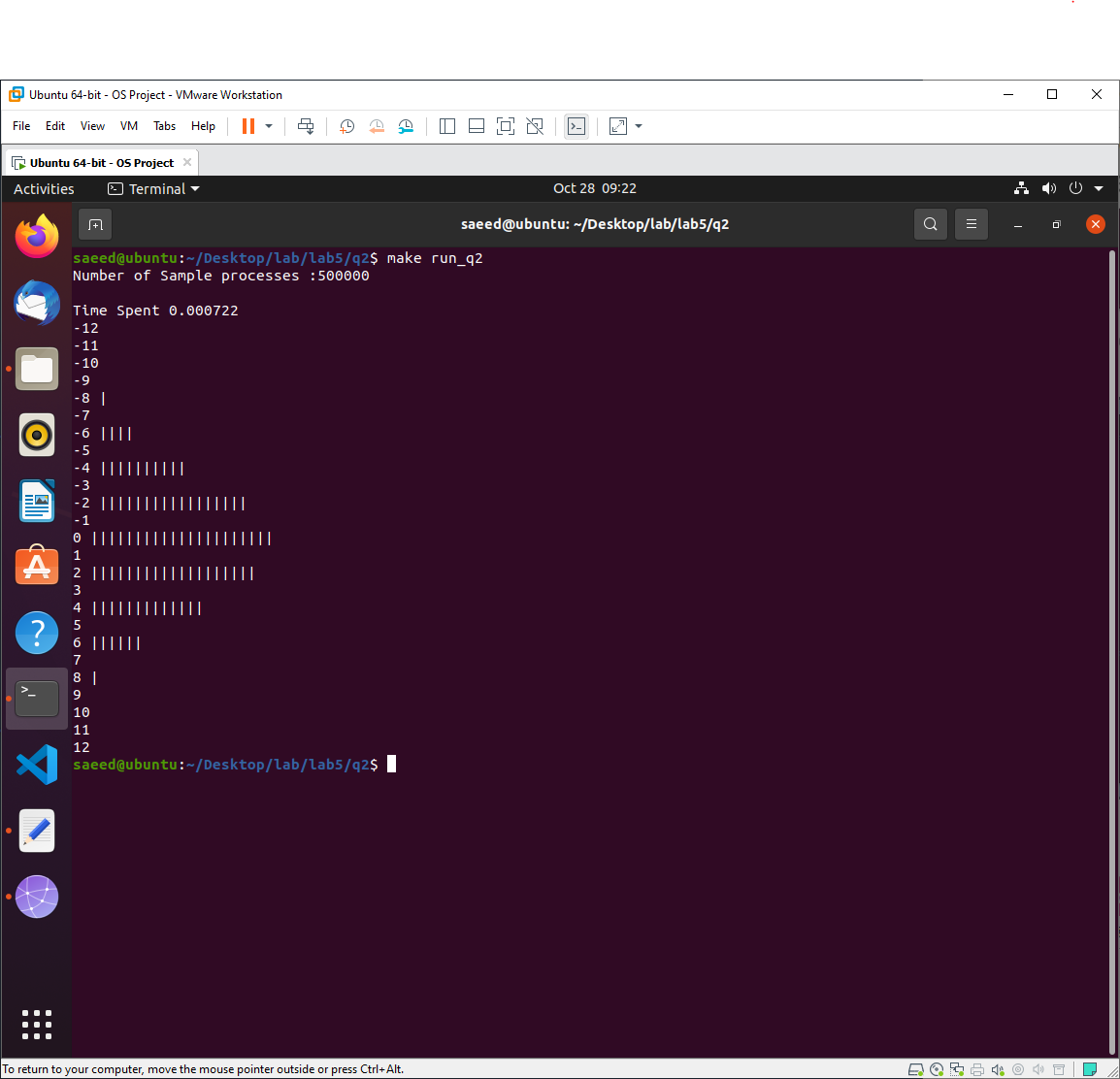
تعداد نمونه 5000:



تعداد نمونه 50000:



تعداد نمونه 500000:



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

بخش سوم:

بله ، زیرا در این برنامه دارای پردازه های مختلفی است که هر کدام می تواند به متغییر های مشترک دسترسی داشته باشند و آن را تغییر دهند. متغییر hist[counter + 12] این متغییر 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 متغییر بین ریسمان های برنامه به اشتراگ گذاشته می شود. روی این متغییر می توان از دو تابع wait و signal استفاده کرد. با صدا زدن تابع wait مقدار آن 1 واحد اضافه می شود و با صدا زدن تابع signal از مقدار آن 1 واحد کاسته می شود. اگر مقدار این متغییر صفر شود ، دیگر نمیتوان روی آن تابع wait را صدا زد. پس پردازه ای که wait را صدا کرده اجرایش متوقف می شود تا زمانی که پردازه 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% |