# به نام خدا

## گزارشکار آزمایش ششم

## درس سیستم عامل

استاد : حسینی تودشکی

سید سعید شفیعی

آبان ۱۴۰۰

## بخش اول:

در این بخش با مسئله مهم Producer/Consumer سروکار داریم. همانطور که در دستور کار توضیح داده شده است، می توان این فرآیند را به طورعادی پیاده سازی کرد. اما این طرز پیاده سازی مشکل دارد. چون ممکن است فرآیند Reader مقداری که مشغول خواندن آن است، آخرین مقدار نوشته شده توسط Writer یا همان Producer نباشد و در هر بار اجرای برنامه، شرایط یکسانی ایجاد نشود و برنامه دچار Race Condition شود.

برای حل مشکل بالا از راه حل سمافور بهره برده ایم. برای این کار باید کتابخانه semaphore.c را اضافه کنیم. برای پیاده سازی از دو سمافور که به طور معمول استفاده می شوند یعنی mutex و data استفاده شده است.

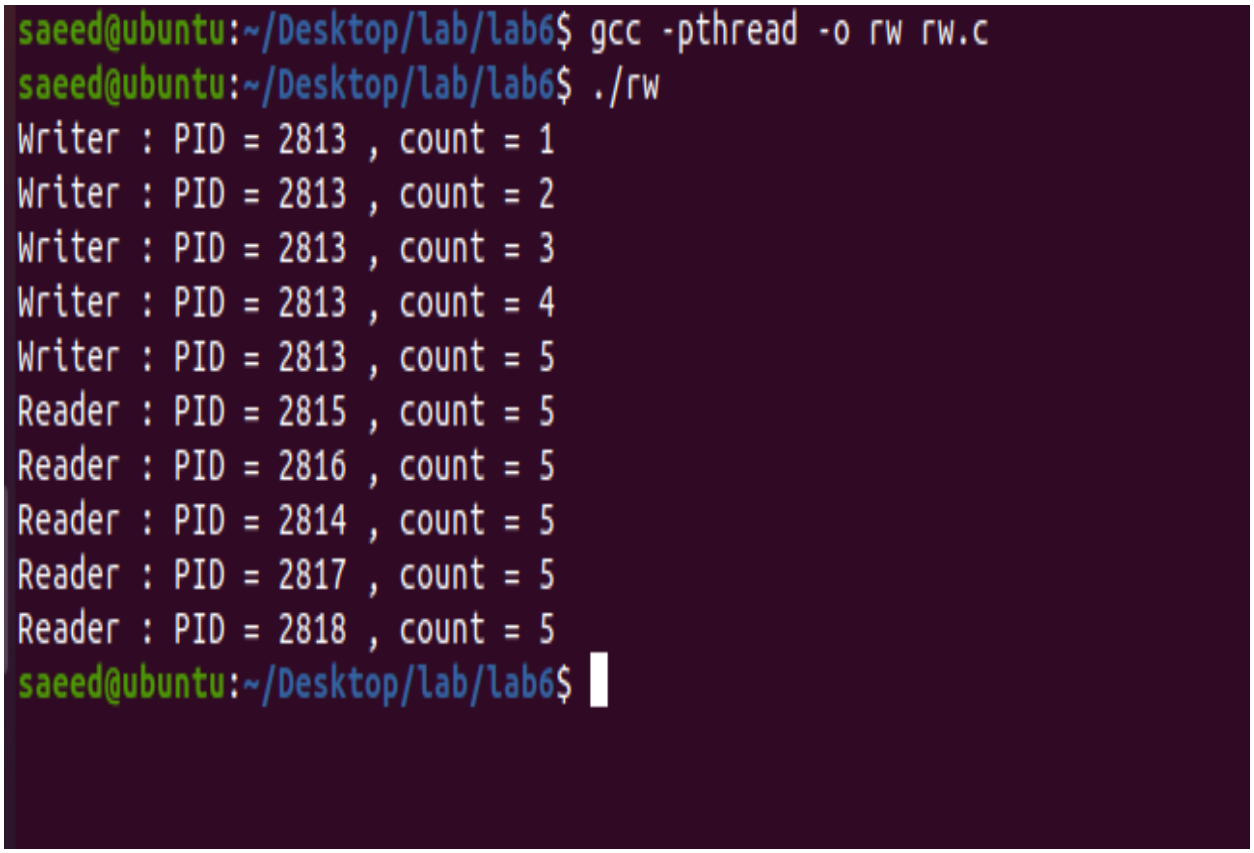
Mutex برای همگام سازی متغییر rc که همان readercounter است.

Data برای همگام سازی reader و writer که در تغییر متغیر count دچار race Condition نشویم.

با استفاده از ()fork پردازه های برای پردازه root ایجاد کرده ایم که تعداد آنها ۱ پردازه برای Writer و پنج پردازه برای reader است. ابتدا reader بر روی سمافور mutex تابع wait را صدا می زند و مقدار آن را صفر می کند. حال اگر اولین reader باشیم باید بر روی سمافور data هم یک wait صدا کنیم و مقدار آن را صفر کنیم تا writer نتواند در حال خواندن مقداری را بر روی حافظه ی مشترک بنویسد. سپس با صدا زدن post رو mutex از حالت بلاک خارج می شویم تا خوانندگان دیگر هم قابلیت خواندن داشته باشند. در انتهای خواندن هم باید روی mutex یک wait صدا بزنیم و مقدار rc را کاهش دهیم.

در صورتی که آخرین reader باشیم، باید writer را از بلاک خارج کنیم تا بتواند داده بنویسد. برای این کار باید روی سمافور data، یک post صدا بزنیم. در انتها هرکدام از reader ها که مقدار rc را کم کردند باید روی سمافور mutex، یک post صدا بزنند تا reader های دیگر بتوانند مقدارrc را تغییر دهند.

نتیجه کامپایل کد موجود در گزارش :



## بخش دوم: Dining Philosophers

در این مسئله Deadlock در شرایطی رخ می دهد که هیچ کدام از پردازه ها در حال اجرا نباشند. یعنی اگر فرض کنیم همه فیلسوف ها همزمان گرسنه شوند، در صورت توزیع درست چنگال ها ماکسیمم دو فیلسوف می توانند غذا بخورند. اما حالتی از تقسیم شدن چنگال ها داریم که هیچ فیلسوفی نتواند غذا بخورد.

این حالت به این صورت است که همه فیلسوف ها ابتدا چنگال سمت راست را بر می دارند و بعد به سراغ چنگال های سمت چپ می روند، در این حالت چنگال چپ توسط فیلسوف کناری برداشته شده است پس هیچ کدام قادر به غذا خوردن نیستند. پس این حالت، حالت Deadlock است.

برای حل این مشکل از mutex استفاده می کنیم که زمانی که فیلسوف چنگال را بر می دارد، چنگال را mutex تعریف کرده و آن را lock می کنیم تا زمانی که غذا خوردن آن برطرف شود و پس از آن ، آن را unlock می کنیم.

سمافور های موجود در این کد ۳ نوع هستند ، ۲ تای آنها برای برداشتن و پایین گذاشتن چنگال و آخری برای خود چنگال. از سمافور های برداشتن و گذاشتن برای این استفاده می کنیم که چنگال چپ و راست به صورت همزمان و طی فرایند Atomic برداشته شوند.

خروجی کد که در صفحه بعد قابل مشاهده است نشان می دهد که حداکثر دو فیلسوف همزمان می توانند غذا بخورند و هیچ حالت Deadlock هم وجود ندارد. این پروسه تا پایان یافتن آن توسط ctrl + c ادامه دارد.

```
saeed@ubuntu:~/Desktop/lab/lab6$ ./dp
Philosopher[0] is thinking
Philosopher[0] is eating using chopstick[0], chopstick[1]
Philosopher[3] is thinking
Philosopher[3] is eating using chopstick[3], chopstick[4]
Philosopher[2] is thinking
Philosopher[0] is finished eating
Philosopher[3] is finished eating
Philosopher[2] is eating using chopstick[2], chopstick[3]
Philosopher[1] is thinking
Philosopher[2] is finished eating
Philosopher[1] is eating using chopstick[1], chopstick[2]
Philosopher[4] is thinking
Philosopher[0] is thinking
Philosopher[4] is eating using chopstick[4], chopstick[0]
Philosopher[4] is finished eating
Philosopher[0] is eating using chopstick[0], chopstick[1]
Philosopher[3] is thinking
Philosopher[3] is eating using chopstick[3], chopstick[4]
Philosopher[1] is finished eating
Philosopher[1] is thinking
Philosopher[0] is finished eating
Philosopher[3] is finished eating
Philosopher[1] is eating using chopstick[1], chopstick[2]
Philosopher[4] is thinking
Philosopher[4] is eating using chopstick[4], chopstick[0]
Philosopher[2] is thinking
Philosopher[4] is finished eating
Philosopher[1] is finished eating
Philosopher[0] is thinking
Philosopher[0] is eating using chopstick[0], chopstick[1]
Philosopher[3] is thinking
Philosopher[2] is eating using chopstick[2], chopstick[3]
Philosopher[0] is finished eating
Philosopher[3] is eating using chopstick[3], chopstick[4]
Philosopher[4] is thinking
Philosopher[2] is finished eating
Philosopher[3] is finished eating
Philosopher[4] is eating using chopstick[4], chopstick[0]
Philosopher[1] is thinking
Philosopher[0] is thinking
Philosopher[1] is eating using chopstick[1], chopstick[2]
Philosopher[4] is finished eating
Philosopher[0] is eating using chopstick[0], chopstick[1]
Philosopher[1] is finished eating
Philosopher[1] is thinking
Philosopher[0] is finished eating
```