دانشگاه علم و صنعت

دانشکده مهندسی کامپیوتر

درجه تحصیلی: کارشناسی

گزارشکار تکلیف3 OS

گردآورنده:

پرنیان شاکریان - 99400064

استاد:

دکتر انتظاری

سال تحصیلی: اردیبهشت ۱۴۰۲

خلاصه:

در تکلیف سوم سیستم عامل قصد داریم از lock و semaphore در حل تمارین استفاده کنیم. اگر بخواهیم به طور خلاصه در مورد ماهیت کدها توضیح دهیم Process، پردازه مجموعه‌ای از کدها، حافظه، داده و سایر منابع است. Thread توالی از کدها به حساب می‌آید که در داخل محدوده یک پردازه اجرا می‌شود. lock یک مکانیسم همگام سازی اولیه است که انحصار متقابل برای منابع مشترک را فراهم می‌کند. هنگامی که یک Thread یک lock می‌گیرد، دسترسی انحصاری به منبع پیدا می‌کند و تا زمانی که lock آزاد نشود، از دسترسی سایر Thread ها به آن جلوگیری می‌کند. lock را می توان با استفاده از الگوریتم های مختلفی مانند mutexes پیاده سازی کرد. سمافور یک مکانیسم همگام سازی پیشرفته تر است که به چندین رشته یا فرآیند اجازه می دهد تا به یک منبع مشترک، به صورت همزمان دسترسی داشته باشند. (تا حداکثر تعداد مشخصی) سمافور از یک شمارنده تشکیل شده است که تعداد منابع موجود را ردیابی می کند و مکانیسم همگام سازی و سیگنال‌دهی را برای رشته‌ها فراهم می کند تا منابع را به دست آورند و آزاد کنند.

شرح کلی:

۱. هر رشته دارای یک شی از نوع pthread_t مرتبط با آن است که شناسه آن را می گوید. (نمی تواند توسط چندین رشته به طور همزمان استفاده شود)
۲. وقتی یک رشته نیاز به یک منبع مشترک دارد، ابتدا باید lock یا سمافور را به دست آورد. پس از اتمام کار، باید آنها را آزاد کرده تا رشته‌های دیگر بتوانند به منبع دسترسی پیدا کنند.
۳. تمام کتابخانه ها و headerهای مورد نیاز ما برای تمرین 3 به طور کلی و برای آسانی کار اینجا نوشته شده است.

```c
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>
#include <semaphore.h>
#include <unistd.h>
```

سوال ۱: حساب مشترک.

در ابتدا کتابخانه‌های مورد نظر خود را include خواهیم کرد. (کتابخانه های مورد نیاز ما برای این تمرین در اول داک نوشته شده است) در ادامه حداکثر تعداد تراکنش ها و افراد را در لیست تراکنش ها مشخص می‌کنیم. میتوان N را هر مقدار که نیاز داریم تعریف کنیم. سپس یک data structure جدید به نام transaction_list_type را تعریف کرده که شامل یک عدد صحیح m، یک آرایه صحیح تراکنش با اندازه N و یک index است. دو خط بعدی متغیرهای سراسری هستند. balance یک عدد صحیح طولانی است که موجودی فعلی را ذخیره می کند. mutex یک متغیر سمافور است که برای همگام‌سازی دسترسی به متغیر balance در بین رشته‌ها استفاده می‌شود. تابع do_transactions آرگومان void pointer را می‌پذیرد. آرگومان arg یک pointer به ساختار transaction_list_type است که فهرست تراکنش ها را برای یک شخص نشان می دهد. این تابع روی تراکنش ها حلقه زده و عملیات زیر را انجام می دهد:

۱. sem_wait: تابع منتظر می ماند تا mutex سمافور در دسترس قرار بگیرد. اگر سمافور در دسترس نباشد، thread را مسدود می کند. اگر عملیات بتواند انجام شود در خط بعدی جزئیات تراکنش در حال اعمال را چاپ می کند.
۲. balance: در بخش critical section موجودی را به روز کرده و موجودی جدید را چاپ میکند. در ادامه که thread به مدت ۲ ثانیه قبل از انجام تراکنش بعدی وارد فاز sleep میشود.
۳. sem_post: با فراخوانی این بخش، تابع به mutex سمافور سیگنال می دهد که critical section کامل شده و سمافور برای استفاده از رشته های دیگر در دسترس است. در آخر پس از اتمام فرایند تابع pthread_exit رشته را خاتمه میدهد.

تابع main ابتدا از کاربر می‌خواهد n و m، تعداد افراد و تعداد تراکنش‌های هر شخص را وارد کند. سپس، آرایه‌ای از transaction_list_type به نام transaction_list و آرایه‌ای از pthread_t به نام threads با اندازه‌های N اعلام می‌کند که در آن N را ابتدای کد ۱۰۰۰ تعریف کرده بودیم. سپس برنامه از یک حلقه تودرتو استفاده می کند تا

از کاربر بخواهد تراکنش ها را برای هر شخص وارد کند سپس ورودی‌های کاربر را در آرایه تراکنش‌های ساختار transaction_list_type وارد کرده و همچنین تعداد تراکنش ها و index هر شخص را در فیلدهای مربوط به ساختار ذخیره میکند. پس از آن، برنامه با استفاده از یک حلقه تو در تو، تراکنش های هر فرد را چاپ می کند. سپس، برنامه سمافور mutex را با استفاده از sem_init مقداردهی اولیه می‌کند. ۰ را به عنوان آرگومان دوم برای مشخص کردن اینکه سمافور بین رشته‌های همان فرآیند به اشتراک گذاشته می‌شود و ۱ را به‌عنوان آرگومان سوم برای مشخص کردن اینکه سمافور در ابتدا قفل است یا خیر ارسال می‌کند. سپس برنامه با استفاده از حلقه و pthread_create، n تعداد رشته ایجاد کرده و بعد از آن آدرس تابع do_transactions و آدرس عنصر آرایهtransaction_list مربوط به رشته فعلی را ارسال می کند. پس از ایجاد تمام رشته‌ها، برنامه منتظر می‌ماند تا با استفاده از حلقه دیگر و pthread_join پایان یابد. در نهایت، برنامه با استفاده از sem_destroy سمافور mutex را از بین می برد و balance نهایی را چاپ می کند.
خروجی‌های زیر را برای ورودی های دلخواه من خواهیم داشت:

Output Clear

```
/tmp/9wpuUzyXwD.o
Please enter n and m in the next line: 1 2
Enter the transactions for person 0.
1 3
this is person 0
1 3
Applying transaction 0 of person 0 : 1
Current balance is : 1
----
Applying transaction 1 of person 0 : 3
Current balance is : 4
----
Final Balance : 4
```

```
/tmp/9wpuUzyXwD.o
Please enter n and m in the next line: 2 2
Enter the transactions for person 0.
1 -3
Enter the transactions for person 1.
4 -1
this is person 0
1 -3
this is person 1
4 -1
Applying transaction 0 of person 0 : 1
Current balance is : 1
----
Applying transaction 1 of person 0 : -3
Current balance is : -2
----
Applying transaction 0 of person 1 : 4
Current balance is : 2
----
Applying transaction 1 of person 1 : -1
Current balance is : 1
----
Final Balance : 1
```