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 ۲: کارخانه دوچرخه سازی.

ابتدا کتابخانه‌های مورد نظر خود را include خواهیم کرد. (کتابخانه های مورد نیاز ما برای این تمرین در اول داک نوشته شده است) همچنین رشته های POSIX و کتابخانه های سمافور اضافه میکنیم. ثابت N برابر ۲۰۰ تعریف می شود. سمافور wheel_sem به عنوان یک متغیر global تعریف می شود. تابع create_wheel که توسط یک thread برای ایجاد چرخ اجرا می شود یک آرگومان void را می گیرد که به یک int فرستاده می شود. (اشاره گر نشان دهنده شماره چرخ است.) داخل تابع، شماره چرخ با استفاده از printf() روی کنسول چاپ می شود تا نشان دهد کدام چرخ در حال ایجاد است. سپس، sem_post را فراخوانی می کند تا نشان دهد که یک چرخ ساخته شده است. در نهایت، تابع NULL را برمی‌گرداند که نشان می‌دهد کامل شده است.

تابع create_bicycle توسط رشته هایی فراخوانی می شود که دوچرخه ها را ایجاد می کنند. تابع یک آرگومان void می گیرد که به یک int فرستاده می شود و نشان دهنده شماره دوچرخه است. در داخل تابع، شماره دوچرخه با استفاده از printf روی کنسول چاپ می شود تا نشان دهد کدام دوچرخه در حال ایجاد است. سپس، نخ به مدت ۲ ثانیه با استفاده از sleep زمان ایجاد اولین چرخ را شبیه‌سازی کند. سپس تابع sem_wait را فراخوانی می‌کند تا منتظر سیگنال‌دهی به سمافور باشد که نشان می‌دهد چرخ ساخته شده است. هنگامی که سمافور علامت داده می شود، تابع پیامی را چاپ می کند که نشان می دهد اولین چرخ به دوچرخه وصل شده است. سپس، دوباره sem_wait را فراخوانی می کند تا منتظر شود تا چرخ دوم ساخته شود. هنگامی که چرخ دوم ساخته شد، تابع پیامی را چاپ می کند که نشان می دهد به دوچرخه متصل شده است. در نهایت تابع NULL را برمی گرداند، که نشان می دهد کامل شده است.

تابع اصلی نقطه ورود برنامه است. با اعلام چند متغیر، از جمله آرایه ای از ساختار pthread_t برای نگه داشتن شناسه رشته و آرایه ای از اعداد صحیح برای نگهداری اعداد چرخ و دوچرخه، شروع می شود. سپس کد از کاربر می‌خواهد تعداد دوچرخه‌ها مورد نظر خود را وارد کند و semaphore wheel_sem را با ۰ مقداردهی اولیه می‌کند. سپس، آرایه num_arr را با اعداد صحیح از ۰ تا ۲*n-1 پر میکند. (n تعداد دوچرخه های وارد

شده توسط کاربر است) و با استفاده از تابع pthread_create رشته هایی را برای ساخت دوچرخه ها و چرخ ها ایجاد می کند. پس از ایجاد تمام رشته ها، تابع اصلی با استفاده از تابع pthread_join منتظر می ماند تا تکمیل شود. هنگامی که تمام رشته ها تکمیل شدند، سمافور با استفاده از تابع sem_destroy از بین می رود و برنامه با مقدار بازگشتی ۰ خارج می شود.

خروجی زیر را برای ورودی دلخواه من خواهیم داشت: