به نام خدا

سیستم های عامل
دکتر ثبوتی
پروژه پایان ترم : سیستم مدیریت حساب کاربران
نیمسال اول 1403-1404
مهلت تحویل : چهارشنبه 03/11/1403

مقدمه

شما باید یک سیستم مدیریت حساب کاربران طراحی کنید که کاربران متعددی به‌طور همزمان به منابع مشترک دسترسی داشته باشند. هدف این پروژه، آشنایی با مفاهیم همگام‌سازی و پیشگیری از شرایط بن‌بست در سیستم‌های چندوظیفه‌ای است.

اهداف پروژه

1. پیاده‌سازی همگام‌سازی: اطمینان از اجرای صحیح و بدون تداخل عملیات مربوط به منابع مشترک.
2. جلوگیری از بن‌بست: طراحی سازوکارهایی برای جلوگیری از وقوع بن‌بست میان کاربران یا فرآیندها.

مشخصات پروژه

1. ساختار پروژه:
   - هر کاربر به عنوان یک فرآیند یا رشته ((**Thread** در سیستم تعریف شود.
   - کاربران می‌توانند عملیات زیر را انجام دهند:
     - ایجاد حساب کاربری: تعریف حساب جدید با موجودی اولیه.
     - بررسی موجودی: مشاهده موجودی فعلی حساب.
     - واریز وجه: اضافه کردن مبلغ مشخص به حساب.
     - برداشت وجه: کاهش مبلغ مشخص از حساب (در صورت کافی بودن موجودی).
     - انتقال وجه: انتقال مبلغ مشخص از یک حساب به حساب دیگر.
     - تاریخچه تراکنش‌ها: لیست تراکنش‌های موفق کاربر نمایش داده شود.
   - تمام عملیات باید با استفاده از فایل انجام شود و وضعیت حساب‌ها در فایل ذخیره شود.
   - تاریخچه تراکنش ها برای هر حساب باید لاگ شود.
2. منابع مشترک:
   - اطلاعات مربوط به حساب‌ها در فایل‌هایی ذخیره می‌شوند.
   - دسترسی همزمان چند کاربر به یک فایل باید به درستی مدیریت شود.
   - عملیات انتقال وجه نیاز به دسترسی همزمان به دو فایل مربوط به دو حساب مختلف دارد.

3. همگام‌سازی و جلوگیری از بن‌بست:
    - از مکانیزم‌های همگام‌سازی مانند **Semaphore** ، **Mutex** یا **Monitor** استفاده شود.
    - برای عدم رخ دادن بن بست باید ایده‌ای پیاده‌سازی شود :
        - از روش‌های مناسب برای جلوگیری از وقوع بن‌بست بهره ببرید، (Deadlock prevention) مانند:
            - ترتیب‌دهی منابع (Resource Ordering).
            - پیشگیری از نگه‌داشتن و انتظار (No Hold and Wait).
        - از روش های مناسب برای اجتناب از بن بست استفاده کنید. (Deadlock avoidance) مانند :
            - الگوریتم بانکداران (Bankers Algorithm)
        - در صورت وقوع بن‌بست، سیستم باید آن را شناسایی کند و فرآیندهای مورد نظر را خاتمه دهد. ((Deadlock detection and recovery

زبان و ابزارهای قابل استفاده

- زبان‌های برنامه‌نویسی:محدودیتی در استفاده از زبان برنامه نویسی نیست. بخش های مربوط به همگام‌سازی چند فرآیند یا رشته باید از طریق Semaphore ، mutex یا Monitorانجام شود. (استفاده از کتابخانه های third party برای همگام‌سازی مجاز نمی باشد.)
- سیستم فایل: استفاده از فایل‌های متنی یا باینری برای ذخیره اطلاعات. استفاده از هر نوع پایگاه داده برای ذخیره داده ها به دلیل پیاده‌سازی سیاست های همگام سازی و جلوگیری از deadlock در پایگاه داده ها مجاز نمی باشد. همچنین استفاده از هرگونه سیاست برای ساختار فایل ها و یا تعداد آنها بلامانع است.

تحویل پروژه

1. کد منبع: فایل‌های کد پروژه به همراه مستندات.
2. مستندات پروژه: مستندات شما باید شامل توضیحات کامل و روشنی درباره موارد زیر باشد:
    a. طراحی سیستم:
        - چگونه ساختار فایل‌ها و داده‌ها را طراحی کرده‌اید؟
        - چرا این طراحی را انتخاب کرده‌اید؟
    b. مکانیزم‌های همگام‌سازی:
        - برای مدیریت دسترسی همزمان به فایل‌ها از چه مکانیزم‌هایی استفاده کرده‌اید؟
        - توضیح دهید که چگونه این مکانیزم‌ها از وقوع شرایط رقابتی (Race Condition) جلوگیری می‌کنند.

c. جلوگیری از بن‌بست:
   - از چه روش‌هایی برای پیشگیری یا اجتناب از بن‌بست استفاده کرده‌اید؟
   - چرا از این روش استفاده کردید.
   - اگر سیستم شما بن‌بستی را شناسایی می‌کند، چگونه فرآیندهای درگیر را خاتمه داده و منابع را آزاد می‌کند؟
d. مدیریت تراکنش‌ها:
   - اگر تراکنش‌ها اتمیک (Atomic) هستند، توضیح دهید که چگونه اتمیک بودن را تضمین کرده‌اید.
   - اگر خطایی در حین اجرای عملیات رخ دهد، چگونه وضعیت سیستم به حالت قبلی بازمی‌گردد؟
e. چالش‌ها و راهکارها:
   - با چه چالش‌هایی روبرو شدید؟
   - چه راه‌حل‌هایی ارائه دادید و چرا؟

3. پروژه در گروه‌های حداکثر دو نفره قابل پیاده‌سازی می باشد. توجه کنید در صورت انجام پروژه به صورت دونفره، انجام نمره اضافه مورد اول اجباری می باشد. همچنین استفاده از **git** برای گروه‌ها اجباری است.

امکانات پیشنهادی (نمره اضافه)

حداکثر نمره اضافه پروژه 20٪ خواهد بود.

1. تراکنش‌های تجزیه ناپذیر (**Atomic Transaction):** (این قابلیت در صورت دو نفره بودن اجباری است.)
   - هر تراکنش باید یا به طور کامل انجام شود یا در صورت وقوع خطا، وضعیت حساب‌ها به حالت اولیه بازگردد.
   - اطلاعات تمامی تراکنش‌ها باید در یک فایل لاگ ذخیره شوند که شامل جزئیات زیر باشد:
     i. نوع عملیات (واریز، برداشت، انتقال، و غیره)
     ii. زمان انجام عملیات
     iii. وضعیت عملیات (موفق یا ناموفق)
2. پیاده‌سازی رابط کاربری ساده برای شبیه‌سازی عملیات کاربران.
3. پیاده‌سازی **IPC – Inter-Process Communication** :
   - ارتباط بین رابط کاربری (چه به‌صورت گرافیکی و چه در حالت ترمینال) با برنامه اصلی از طریق ipc باشد.
4. هرگونه ایده خلاقانه دیگر با تایید تیم حل تمرین بلامانع است.

توضیحات تکمیلی برای درک بهتر پروژه

پروژه مدیریت حساب کاربران در سیستم‌های چندوظیفه‌ای به شما کمک می‌کند تا با مفاهیم همگام‌سازی (Synchronization) و جلوگیری از بن‌بست (Deadlock Prevention) در برنامه‌های چندوظیفه‌ای آشنا شوید. در این سیستم، چندین فرآیند یا رشته (Threads/Processes) به طور همزمان در حال انجام عملیات روی منابع مشترک هستند و باید اطمینان حاصل شود که هیچ‌گونه تداخلی در اجرای عملیات‌ها به وجود نیاید.

1. منابع مشترک و دسترسی همزمان

در این پروژه، حساب‌های کاربری و تراکنش‌های بانکی منابع مشترک هستند. هر کاربر می‌تواند یک عملیات مانند واریز وجه، برداشت وجه، یا انتقال وجه انجام دهد. چون چندین کاربر ممکن است به طور همزمان به این منابع دسترسی داشته باشند، باید اطمینان حاصل شود که عملیات‌های آن‌ها به درستی همگام‌سازی شوند و تداخل نداشته باشند. به عنوان مثال، ممکن است دو کاربر همزمان بخواهند از موجودی یک حساب برداشت کنند. در این صورت، سیستم باید مطمئن شود که هر عملیات به طور مستقل و دقیق انجام شود و موجودی حساب‌ها به اشتباه تغییر نکند.

2. چالش‌های همگام‌سازی

هنگامی که چندین فرآیند یا رشته به منابع مشترک دسترسی دارند، خطر تداخل یا رقابت منابع (Race Condition) وجود دارد. این به این معنی است که عملیات‌هایی که به طور همزمان اجرا می‌شوند، ممکن است بر یکدیگر تاثیر بگذارند. برای مثال، اگر دو کاربر همزمان بخواهند از یک حساب برداشت کنند، باید سیستم مطمئن شود که موجودی حساب دو بار کاهش نمی‌یابد.

یکی از روش‌های حل این مشکل، استفاده از همگام‌سازی است. این مکانیزم‌ها به برنامه کمک می‌کنند تا دسترسی به منابع مشترک را کنترل کرده و از رقابت منابع جلوگیری کنند. به عبارت دیگر، همگام‌سازی باعث می‌شود که تنها یک فرآیند یا رشته بتواند در هر لحظه به یک منبع مشترک دسترسی پیدا کند.

3. جلوگیری از بن‌بست (**Deadlock**)

بن‌بست زمانی رخ می‌دهد که دو یا چند فرآیند به منابع مختلفی نیاز دارند و هیچ‌کدام نمی‌توانند ادامه دهند زیرا هر یک منتظر دسترسی به منابع دیگری هستند که توسط فرآیندهای دیگر قفل شده‌اند. به عنوان مثال، فرض کنید دو حساب کاربری وجود دارد و دو کاربر همزمان می‌خواهند مبلغی از حساب خود به حساب دیگری منتقل کنند. اگر هر کاربر حساب خود را قفل کرده باشد و منتظر قفل شدن حساب دیگری باشد، هیچ‌کدام قادر به انجام عملیات نخواهند بود و بن‌بست رخ خواهد داد.

4. لاگ‌گیری و تاریخچه تراکنش‌ها

در سیستم‌های چندوظیفه‌ای، داشتن یک سیستم لاگ که تمام تراکنش‌ها را ثبت کند، بسیار مهم است. این کار به دو دلیل انجام می‌شود:

- ردیابی و پیگیری تراکنش‌ها: در صورتی که مشکلی پیش بیاید، می‌توان به راحتی علت مشکل را شناسایی کرد.
- اطمینان از صحت سیستم: تاریخچه تراکنش‌ها به شما کمک می‌کند تا مطمئن شوید که تمام عملیات به درستی انجام شده‌اند و هیچ مشکلی در فرآیند همگام‌سازی وجود ندارد.

5. مفهوم تراکنش‌های اتمیک (**Atomic Transactions**):

تراکنش‌های اتمیک به مجموعه‌ای از عملیات گفته می‌شود که باید به صورت کامل و تمام و کمال انجام شوند یا هیچ‌کدام از آن‌ها اجرا نشوند. به عبارت دیگر، تراکنش‌های اتمیک ویژگی‌های زیر را دارند:

- تمام یا هیچ (**All-or-Nothing**): در صورتی که تراکنش در حین انجام با مشکلی مواجه شود، تمامی تغییرات انجام شده باید به حالت اولیه بازگردند. بدین ترتیب، هیچ‌یک از عملیات‌های آن نباید به صورت نیمه‌کاره یا ناقص باقی بماند.
- غیرقابل تقسیم بودن (**Indivisible**): تراکنش‌ها به عنوان یک واحد واحد عمل می‌کنند و هیچ‌کدام از بخش‌های آن نباید به صورت جداگانه یا در مراحل مختلف اجرا شوند. تمامی عملیات‌ها باید به طور یکجا و در یک زمان انجام شوند.

به طور مثال، اگر یک تراکنش شامل دو عملیات باشد، مانند واریز وجه به حساب A و برداشت از حساب B، در صورتی که برداشت از حساب B موفق باشد اما واریز به حساب A با خطا مواجه شود، سیستم باید تضمین کند که برداشت از حساب B نیز لغو شود. بدین ترتیب، تراکنش به صورت کامل یا اصلاً انجام نمی‌شود.

اهمیت:

در پروژه‌ای که در حال توسعه آن هستید، که به طور خاص به عملیات مالی و مدیریت حساب‌های کاربران مربوط می‌شود، رعایت ویژگی‌های تراکنش‌های اتمیک از اهمیت بسیاری برخوردار است. این عملیات‌ها از جمله واریز وجه، برداشت وجه، و انتقال وجه به حساب‌های مختلف انجام می‌شود و خطا در هر یک از این عملیات‌ها می‌تواند منجر به مشکلات جدی و اشتباهات مالی شود. به عبارت دیگر، در صورت بروز هرگونه نقص یا خطا در یکی از مراحل تراکنش، سیستم باید قادر باشد که تمامی تغییرات را بازگرداند تا وضعیت حساب‌ها به درستی و بدون تناقض باقی بماند.

چرا تراکنش‌ها باید اتمیک باشند؟

دقت در عملیات‌های مالی: اگر تراکنش‌ها اتمیک نباشند، ممکن است عملیات ناقص یا نادرست انجام شود. به عنوان مثال، در سیستم بانکی که در این پروژه پیاده‌سازی می‌شود، اگر بخواهید مبلغی را از یک حساب به حساب دیگر منتقل کنید، باید اطمینان حاصل کنید که همزمان از حساب اول برداشت می‌شود و به حساب دوم واریز می‌شود. اگر تنها یکی از این عملیات‌ها موفقیت‌آمیز باشد، سیستم دچار اختلال خواهد شد و موجودی حساب‌ها نادرست خواهد بود.

حفظ انسجام داده‌ها (**Data Consistency**): تراکنش‌های اتمیک باعث می‌شوند که داده‌ها همیشه در یک وضعیت پایدار و صحیح باقی بمانند. در صورتی که تراکنش‌ها به صورت اتمیک پیاده‌سازی نشوند، داده‌ها ممکن است به صورت ناقص ذخیره شوند که منجر به ناهماهنگی و خطاهای سیستماتیک خواهد شد.

پیشگیری از ناهماهنگی‌ها و خطاهای احتمالی: بدون تراکنش‌های اتمیک، احتمال وقوع ناهماهنگی در داده‌ها بیشتر می‌شود. این ناهماهنگی‌ها ممکن است منجر به بروز مشکلاتی مانند تضاد در داده‌ها و خطاهای اجرایی شوند که می‌تواند بر عملکرد کلی سیستم تاثیر منفی بگذارد.

تضمین صحت و اعتماد پذیری عملیات‌ها: در پروژه شما که تعدادی کاربر به طور همزمان به منابع دسترسی دارند، باید اطمینان حاصل شود که تمامی تراکنش‌ها به درستی و بدون خطا اجرا می‌شوند. تراکنش‌های اتمیک اطمینان می‌دهند که در صورت بروز خطا، هیچ تغییر نادرستی در وضعیت حساب‌ها یا دیگر منابع سیستم ایجاد نخواهد شد.

مثال کاربردی برای درک بهتر:

فرض کنید که دو کاربر به طور همزمان قصد دارند از حساب‌های خود برداشت کنند و پس از آن، مبلغی را به حساب‌های دیگری واریز کنند. در صورتی که در هنگام انجام این عملیات‌ها خطا یا وقفه‌ای ایجاد شود، باید سیستمی وجود داشته باشد که تمامی عملیات‌های قبلی را لغو کند تا از بروز مشکلاتی مانند موجودی‌های نادرست یا تراکنش‌های ناقص جلوگیری شود. تراکنش‌های اتمیک به این اطمینان کمک می‌کنند که هیچ عملیات ناقص یا اشتباهی در سیستم باقی نماند.

موفق باشید