دانشگاه صنعتی خواجه نصیرالدین طوسی

پروژه پایانی درس سیستم های عامل

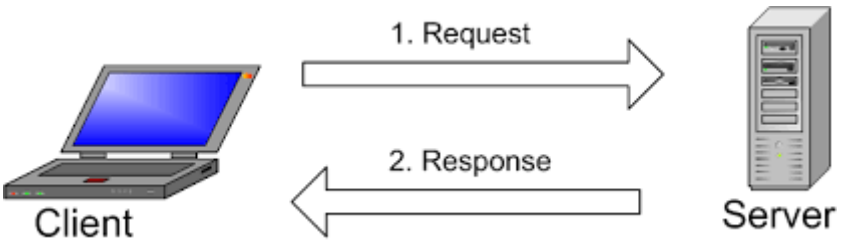
سرور فایل چندنخی با همگام‌سازی پایه

دکتر حامد خانمیرزا

بهار 1403

مرور کلی

در این پروژه، شما یک سرور فایل چندنخی طراحی و پیاده‌سازی خواهید کرد که درخواست‌های متعدد مشتریان برای خواندن و نوشتن فایل‌ها را به‌صورت همزمان مدیریت می‌کند. شما تکنیک‌های پایه همگام‌سازی را برای مدیریت دسترسی به منابع مشترک و اجتناب از شرایط رقابتی خواهید آموخت و اعمال خواهید کرد. این پروژه به شما کمک می‌کند تا مفاهیم کلیدی سیستم‌های عامل مانند مدیریت فرآیندها، ورودی/خروجی فایل و مکانیزم‌های همگام‌سازی را درک کنید.



اهداف پروژه

تا پایان این پروژه، شما باید بتوانید:

۱ . پیاده‌سازی چندنخی در یک زبان برنامه‌نویسی.

۲ . اعمال تکنیک‌های پایه همگام‌سازی با استفاده از mutexها.

۳ . توسعه یک سرور فایل عملیاتی که بتواند درخواست‌های همزمان مشتریان را مدیریت کند.

۴ . مستندسازی فرآیند طراحی و پیاده‌سازی خود.

اجزای پروژه

۱ .سرور فایل چندنخی

- پیاده‌سازی یک سرور TCP پایه که اتصالات مشتریان را قبول می‌کند.
- استفاده از چندنخی برای مدیریت همزمان اتصالات مشتریان متعدد.

۲ . مخزن نخی (Thread Pool)

- پیاده‌سازی یک مخزن نخی برای مدیریت تعداد ثابتی از نخ‌ها برای پردازش درخواست‌های مشتریان.
- اطمینان از استفاده بهینه از منابع سیستم و کاهش سربار ایجاد و از بین بردن نخ‌ها.

۳ . مکانیزم‌های همگام‌سازی

- استفاده از mutexها برای محافظت از بخش‌های بحرانی کد که منابع مشترک (مانند فایل‌ها) در آن‌ها دسترسی دارند.
- اطمینان از همگام‌سازی مناسب برای اجتناب از شرایط رقابتی و تضمین سازگاری داده‌ها.

۴ . عملیات فایل

- پیاده‌سازی عملیات خواندن و نوشتن فایل که توسط مشتریان درخواست می‌شود.
- مدیریت همزمان چندین مشتری که به صورت همزمان فایل‌ها را می‌خوانند و می‌نویسند.

۵ . آزمون و مستندسازی

- توسعه موارد آزمون برای اعتبارسنجی عملکرد سرور فایل.

- گزارش از طراحی و پیاده‌سازی سرور فایل.

زبان‌های برنامه‌نویسی پیشنهادی

- Java
- C

مراحل پروژه

مرحله ۱: طراحی و پیاده‌سازی سرور پایه

- سرور پایه : پیاده‌سازی یک سرور TCP پایه که اتصالات مشتریان را قبول می‌کند. تست با یک مشتری ساده برای اطمینان از اینکه سرور اتصالات را قبول می‌کند.

مرحله ۲: چندنخی و مخزن نخی

- چندنخی : افزودن چندنخی برای مدیریت همزمان اتصالات مشتریان متعدد. ایجاد نخ‌ها برای هر اتصال مشتری.
- مخزن نخی : پیاده‌سازی یک مخزن نخی ساده برای مدیریت تعداد ثابتی از نخ‌ها برای پردازش درخواست‌های مشتریان. اطمینان از استفاده بهینه از نخ‌ها.

مرحله ۳: همگام‌سازی و عملیات فایل

- همگام‌سازی : افزودن mutexها برای محافظت از بخش‌های بحرانی و اطمینان از همگام‌سازی مناسب. تمرکز بر منابع مشترک مانند دسترسی به فایل.
- عملیات فایل : پیاده‌سازی عملیات خواندن و نوشتن فایل. اطمینان از اینکه مشتریان می‌توانند همزمان از فایل‌ها بخوانند و بنویسند بدون خرابی داده‌ها.

مرحله ۴: آزمون و مستندسازی

- آزمون : توسعه و اجرای موارد آزمون برای اطمینان از عملکرد صحیح سرور در سناریوهای مختلف. تست برای مسائل همزمانی و سازگاری داده‌ها.
- مستندسازی : گزارشی از functionality سیستمی که طراحی کردید.

دستورالعمل‌های دقیق

مرحله ۱: طراحی و پیاده‌سازی سرور پایه

۱ . پیاده‌سازی سرور پایه
- یک زبان برنامه‌نویسی (جاوا یا سی) را انتخاب کنید.
- پیاده‌سازی یک سرور TCP که به اتصالات مشتریان گوش می‌دهد.
- نوشتن یک مشتری ساده که به سرور متصل شده و یک پیام آزمایشی ارسال می‌کند.
- اطمینان حاصل کنید که سرور اتصال را قبول کرده و پیام را به درستی مدیریت می‌کند.

مرحله ۲: چندنخی و مخزن نخی

۱ . چندنخی
- سرور را طوری تغییر دهید که برای هر اتصال مشتری یک نخ جدید ایجاد کند.
- اطمینان حاصل کنید که سرور می‌تواند اتصالات مشتریان متعدد را همزمان مدیریت کند.
- سرور را با چندین مشتری تست کنید تا مطمئن شوید که به درستی کار می‌کند.

۲ . مخزن نخی
- پیاده‌سازی یک مخزن نخی برای مدیریت تعداد ثابتی از نخ‌ها.
- استفاده مجدد از نخ‌ها برای پردازش درخواست‌های مشتریان به‌منظور کاهش سربار ایجاد و از بین بردن نخ‌ها.
- اطمینان از تخصیص پویا نخ‌ها بر اساس بار کاری.

مرحله ۳: همگام‌سازی و عملیات فایل

۱ . همگام‌سازی
- بخش‌های بحرانی در کد خود را که منابع مشترک در آن‌ها دسترسی دارند شناسایی کنید (مانند عملیات فایل).

- پیاده‌سازی mutexها برای محافظت از این بخش‌های بحرانی.
- اطمینان از همگام‌سازی مناسب برای اجتناب از شرایط رقابتی و تضمین سازگاری داده‌ها.

۲ . عملیات فایل

- پیاده‌سازی عملیات خواندن و نوشتن فایل که توسط مشتریان درخواست می‌شود.
- اطمینان از اینکه مشتریان می‌توانند همزمان از فایل‌ها بخوانند و بنویسند.
- مدیریت قفل‌گذاری و بازکردن قفل فایل به‌طور مناسب برای حفظ یکپارچگی داده‌ها.

مرحله ۴: آزمون، مستندسازی و ارائه

۱ . آزمون

- توسعه موارد آزمون جامع برای اعتبارسنجی عملکرد سرور فایل.
- تست سرور در سناریوهای مختلف، شامل مشتریان همزمان متعدد، بررسی ایجاد نشدن مشکل هنگام نوشتن همزمان در یک فایل.
- اطمینان از اینکه سرور سازگاری داده‌ها را حفظ می‌کند و مسائل همزمانی را به درستی مدیریت می‌کند.

۲ . مستندسازی

- نوشتن گزارش برای پروژه خود.
- گزارشی از functionality سیستمی که طراحی کردید.

معیارهای ارزیابی

- **درستی و توانمندی** : سرور فایل باید به درستی کار کرده و در سناریوهای مختلف توانمندی داشته باشد.
- **استفاده مؤثر از چندنخی و همگام‌سازی** : نشان دادن استفاده صحیح از چندنخی و مکانیزم‌های همگام‌سازی برای مدیریت درخواست‌های همزمان مشتریان و اجتناب از شرایط رقابتی.
- **درک مفاهیم سیستم‌های عامل** : نشان دادن درک واضح از مدیریت فرآیندها، ورودی/خروجی فایل و تکنیک‌های همگام‌سازی.

بخش امتیازی:

اکنون تصور کنید که این TCP Server یک سرور برای سرویس‌دهی به ساعت‌های هوشمند است؛ یعنی ساعت‌های هوشمند پیام‌های خود را به این سرور ارسال می‌کنند.
تصور کنید که ۴ ساعت هوشمند از خانواده fa95s داریم که functionality این ساعت‌ها بدین شرح است:

۱. دستور poweroff: این دستور از سمت سرور به ساعت ارسال می‌شود و پس از ارسال این دستور، ساعت خاموش می‌شود.

فرمت پیام: [3G*IMEI*POWEROFF]

۲. دستور find: این دستور هم مشابه دستور قبلی از سمت سرور به ساعت ارسال می‌شود. این دستور زمانی بکار می‌رود که ساعت گم شده‌باشد. با ارسال این دستور به ساعت،ساعت به مدت ۳۰ ثانیه شروع به زنگ زدن می‌کند.

فرمت پیام: [3G*IMEI*FIND]

۳. پیام health data: ساعت در هر ۳۰ ثانیه باید اطلاعات سلامتی را به سرور ارسال بکند.

فرمت پیام:
[3G*IMEI*HEALTH*heartrate, blood pressure low, blood pressure high]

۴. پیام location: ساعت‌ها در هر ۴۵ ثانیه مکان جغرافیایی (latitude و longitude) خود را به سرور ارسال می‌کنند.

فرمت پیام: [3G*IMEI*UD, lat, lon]

نکاتی که باید در طراحی ساعت‌ها و server رعایت شوند:

- **فقط یک** TCP server باید وجود داشته باشد که به چندین ساعت سرویس می‌دهد.
- طراحی watch ها باید بگونه‌ای باشد که functionality های گفته‌شده لحاظ شود.
- IMEI یک کد ۱۰ رقمی **یکتا** است که هر کارخانه‌ای پس از ساخت محصول خود، به آن اختصاص می‌دهد.
- برای شبیه‌سازی ساعت‌ها می توانید ۴ تا فایل client پیاده‌سازی کنید که به سرور متصل می‌شوند؛ در واقع، هر client file شبیه‌سازی یک ساعت هوشمند خواهدبود.
- از آنجا که فقط یک سرور داریم، سرویس‌دهی سرور به ۴ ساعت fa95s، باید بصورت **همروند** پیاده‌سازی شود؛ یعنی پس از ایجاد connection از سمت ساعت، بهتر است یک thread ایجاد شود تا task های لازم را انجام دهد.
- یک فایل log هم نیاز است تا فعالیت‌های ساعت و سرور در آن نوشته‌شود. دقت کنید که در این فایل، فعالیت همه ساعت‌ها نوشته‌می‌شود و تمهیدات لازم (lock) **باید** اندیشیده‌شود.