به نام خدا

آزمایشگاه سیستم عامل

پروژه چهارم: همگام سازی

طراحان: علی عباسی، علی عطاالله‌ی

مقدمه

در این پروژه با سازوکارهای همگام‌سازی[1] سیستم عامل‌ها آشنا خواهید شد. با توجه به این که سیستم عامل xv6 از ریسه‌های سطح کاربر پشتیبانی نمی‌کند، همگام‌سازی در سطح پردازه‌ها مطرح خواهد بود. همچنین به علت عدم پشتیبانی از حافظه مشترک در این سیستم عامل، همگام‌سازی در سطح هسته صورت خواهد گرفت. به همین سبب مختصری راجع به این قسم از همگام‌سازی توضیح داده خواهد شد.

[1] Synchronization Mechanisms

ضرورت همگام سازی در هسته سیستم عامل ها

هسته سیستم عامل‌ها دارای مسیرهای کنترلی[2] مختلفی میباشد. به طور کلی، دنباله دستورالعمل‌های اجرا شده توسط هسته جهت مدیریت فراخوانی سیستمی، وقفه یا استثنا این مسیرها را تشکیل میدهند. در این میان برخی از سیستم عامل‌ها دارای هسته با ورود مجدد[3] می باشند. بدین معنی که مسیرهای کنترلی این هسته‌ها قابلیت اجرای همروند[4] دارند. تمامی سیستم عامل های مدرن کنونی این قابلیت را دارند. مثلا ممکن است برنامه سطح کاربر در میانه اجرای فراخوانی سیستمی در هسته باشد که وقفه هایی رخ دهد. به این ترتیب در حین اجرای یک مسیر کنترلی در هسته (اجرای کد فراخوانی سیستمی)، مسیر کنترلی دیگری در هسته (اجرای کد مدیریت وقفه) شروع به اجرا نموده و به نوعی دوباره ورود به هسته صورت می پذیرد. وجود همزمان چند مسیر کنترلی در هسته می‌تواند منجر به وجود شرایط مسابقه برای دسترسی به حالت مشترک هسته گردد. به این ترتیب، اجرای صحیح کد هسته مستلزم همگام سازی مناسب است. در این همگام سازی باید ماهیتهای مختلف کدهای اجرایی هسته لحاظ گردد.

هر مسیر کنترلی هسته در یک متن خاص اجرا می‌گردد. اگر کد هسته به طور مستقیم یا غیرمستقیم توسط برنامه سطح کاربر اجرا گردد، در متن پردازه[5] اجرا می‌گردد. در حالی که کدی که در نتیجه وقفه اجرا می‌گردد در متن وقفه[6] است. به این ترتیب فراخوانی سیستمی و استثناها در متن پردازه فراخواننده هستند. در حالی که وقفه در متن وقفه اجرا میگردد. به طور کلی در سیستم عامل‌ها کدهای وقفه قابل مسدود شدن نیستند. ماهیت این کدهای اجرایی به این صورت است که باید در اسرع وقت اجرا شده و لذا قابل زمانبندی توسط زمانبند نیز نیستند. به این ترتیب سازوکار همگام‌سازی آنها نباید منجر به مسدود شدن آنها گردد، مثلا از قفلهای چرخشی[7] استفاده گردد یا در پردازنده های تک هسته ای وقفه غیر فعال گردد.

[2] Control Path
[3] Reentrant Kernel
[4] Concurrent
[5] Process Context
[6] Interrupt Context
[7] Spin Locks

همگام‌سازی در xv6

قفل گذاری در هسته xv6 توسط دو سری تابع صورت می‌گیرد. دسته اول شامل توابع acquire (خط 1573) و release (خط 1601) می‌شود که یک پیاده‌سازی ساده از قفل‌های چرخشی هستند. این قفل‌ها منجر به انتظار مشغول[8] شده و در حین اجرای ناحیه بحرانی وقفه را نیز غیرفعال می‌کنند.

1) علت غیرفعال کردن وقفه در هنگام استفاده از این نوع قفل چیست؟ چرا ممکن است CPU با مشکل deadlock رو به رو شود؟
2) توابع pushcli و popcli به چه منظور استفاده شده و چه تفاوتی با cli و sti دارند؟
3) چرا قفل مذکور در سیستم های تک هسته ای مناسب نیست؟ روی کد توضیح دهید.
4) در مجموعه دستورات RISC-V، دستوری با نام amoswap وجود دارد. دلیل تعریف و نحوه کار آن را توضیح دهید.

دسته دوم شامل توابع acquiresleep (خط 4621) و releasesleep (خط 4633) بوده که مشکل انتظار مشغول را حل نموده و امکان تعامل میان پردازه‌ها را نیز فراهم می‌کنند. تفاوت اصلی توابع این دسته نسبت به دسته قبل این است که در صورت عدم امکان در اختیار گرفتن قفل، از تلاش دست کشیده و پردازنده را رها می‌کنند.

5) مختصری راجع به تعامل میان پردازه‌ها توسط دو تابع مذکور توضیح دهید.
6) حالات مختلف پردازه‌ها در xv6 را توضیح دهید. تابع sched چه وظیفه ای دارد؟

یک مشکل در توابع دسته دوم عدم وجود نگهدارنده[9] قفل است. به این ترتیب حتی پردازهای که قفل را در اختیار ندارد می‌تواند با فراخوانی تابع releasesleep قفل را آزاد نماید.

7) تغییری در توابع دسته دوم داده تا تنها پردازه صاحب قفل، قادر به آزادسازی آن باشد. قفل معادل در هسته لینوکس را به طور مختصر معرفی نمایید.

[8] Busy Waiting
[9] Owner

8) روشی دیگر برای نوشتن برنامه‌ها استفاده از الگوریتم های lock-free است. مختصری راجع به آن ها توضیح داده و از مزایا و معایب آن‌ها نسبت به برنامه نویسی با lock بگویید.

پیاده‌سازی متغیرهای مختص هر هسته پردازنده

یکی از روش‌های افزایش کارایی در پردازنده‌ها استفاده از حافظه نهان[10] است. حافظه‌های نهان در سطوح[11] مختلف وجود داشته و می‌توانند محلی[12] یا مشترک[13] باشند. به عنوان مثال، معمولا حافظه نهان سطح یک[14] مختص هر هسته پردازنده بوده و لذا محلی است. بدین ترتیب هرگاه پردازه‌ای در یک متغیر حافظه بنویسد، مقادیر نگه‌داری شده برای این متغیر در حافظه‌های نهان سطح دیگر هسته‌های پردازنده را نامعتبر[15] می‌نماید.

الف) روشی جهت حل این مشکل در سطح سخت‌افزار وجود دارد. مختصرا آن را توضیح دهید.

معتبرسازی مقادیر حافظه نهان محلی، سربار قابل توجهی داشته و می‌تواند کارایی سیستم را پایین بیاورد.

ب) همان‌طور که در اسلایدهای معرفی پروژه ذکر شده‌است، یکی از روش‌های همگام‌سازی استفاده از قفل‌هایی مرسوم به قفل بلیت است. این قفل‌ها را از منظر مشکل مذکور در بالا بررسی نمایید.

در بسیاری از کاربردها می‌توان با استفاده از متغیرهای مختص هر هسته، مشکل را حل نمود. به این ترتیب که به جز در موارد ضروری، دسترسی و به‌روزرسانی را در نسخه مختص هسته جاری از متغیر انجام می‌دهند. بدین ترتیب با کاهش تعداد معتبرسازی، سربار کاهش می‌یابد.

ج) چگونه می‌توان در لینوکس داده‌های مختص هر هسته را در زمان کامپایل تعریف نمود؟

با استفاده از این روش، یک فراخوانی سیستمی تعریف نمایید که تعداد فراخوانی‌های سیستمی اجرا شده در یک بار کاری را روی یک سیستم چهار هسته‌ای برمی‌گرداند.

---

[10] Cache
[11] Levels
[12] Local
[13] Shared
[14] L1 Cache
[15] Invalid

باید به تعداد کافی پردازه ایجاد نمایید که در فایل‌هایی می‌نویسند. همچنین جهت اطمینان از صحت عملکرد باید یک نسخه مشترک میان همه هسته‌ها تعریف شده و با مقدار برگشتی مقایسه گردد. دقت کنید همواره باید متغیر مربوط به هسته در حال اجرا، به‌روزرسانی گردد (راهنمایی: روش پیاده‌سازی قفل چرخشی xv6 می‌تواند راه‌گشا باشد).

نیازی به ترازبندی حافظه نهان[16] نیست.

پیاده‌سازی سازوکار همگام‌سازی با قابلیت اولویت‌دادن

در این قسمت می‌خواهیم سازوکاری برای همگام‌سازی پیاده‌سازی کنیم که در آن پردازه‌ها می‌توانند برای ورود به ناحیه بحرانی دارای اولویت باشند. حال پردازه‌ها می‌توانند برای ورود به ناحیه بحرانی درخواست دهند و پردازه با اولویت بالاتر ابتدا وارد ناحیه بحرانی می‌شود. بعد از آن نیز پرداژه‌ها که در یک صف دارای اولویت هستند به ترتیب وارد می‌شوند. در زمانی که یک پردازه قفل را در اختیار دارد و در ناحیه بحرانی هست نیز ممکن است پردازه‌های جدیدی با اولویت‌های متفاوت وارد شوند که در این صورت ترتیب صف اولویت به صورت پویا عوض می‌شود.

برای پیاده‌سازی فرض کنید که اولویت پردازه‌های متفاوت شماره پردازه آن می‌باشد. شما نیاز است که برنامه سطح کاربری بنویسید که درستی پیاده‌سازی را نشان دهد. برای این کار چند پردازه در سطح کاربر ایجاد کنید که هر پردازه به دنبال ورود به ناحیه بحرانی و دریافت قفل می‌باشد.

حال در ناحیه بحرانی یک کار زمان‌بر انجام دهید تا پردازه‌های دیگر خود را به صف اضافه کنند. در هر مرحله نمایش دهید که پردازه با چه اولویتی وارد ناحیه بحرانی شد. همچنین صف اولویت را نیز نمایش دهید.

[16] Cache Alignment

آیا این پیاده‌سازی ممکن است که دچار گرسنگی شود؟ راه‌حلی برای برطرف کردن این مشکل ارائه دهید. روش ارائه‌شده توسط شما باید بتواند شرایطی را که قفل‌ها دارای اولویت یکسان می‌باشند را نیز پوشش دهد (نیازی به پیاده‌سازی برای این قسمت نیست).

یک نوع پیاده‌سازی همگام‌سازی توسط قفل بلیت[17] انجام می‌شود. آن را بررسی کنید و تفاوت‌های آن با روش همگام‌سازی بالا را بیان کنید.

سایر نکات:

- مدیریت حافظه مناسب در پروژه از نکات مهم پیاده‌سازی است.
- از لاگ‌های مناسب در پیاده سازی استفاده نمایید تا تست و اشکال‌زدایی کد ساده‌تر شود. واضح است که استفاده بیش از حد از آنها باعث سردرگمی خواهد شد.
- کدهای خود را مشابه پروژه‌های پیشین در Github یا Gitlab بارگذاری نموده و آدرس مخزن، شناسه آخرین Commit و گزارش پروژه را در سایت بارگذاری نمایید.
- همه افراد باید به پروژه مسلط باشند و نمره تمامی اعضای گروه لزوما یکسان نخواهد بود.
- در صورت تشخیص تقلب، نمره هر دو گروه صفر در نظر گرفته خواهد شد.
- فصل 4 و انتهای فصل 5 کتاب xv6 می‌تواند مفید باشد.
- هرگونه سوال در مورد پروژه را از طریق ایمیل‌های طراحان می‌توانید مطرح نمایید.

aliabbasi806@gmail.com

aliataollahi40@gmail.com

موفق باشید

[17] Ticket Lock