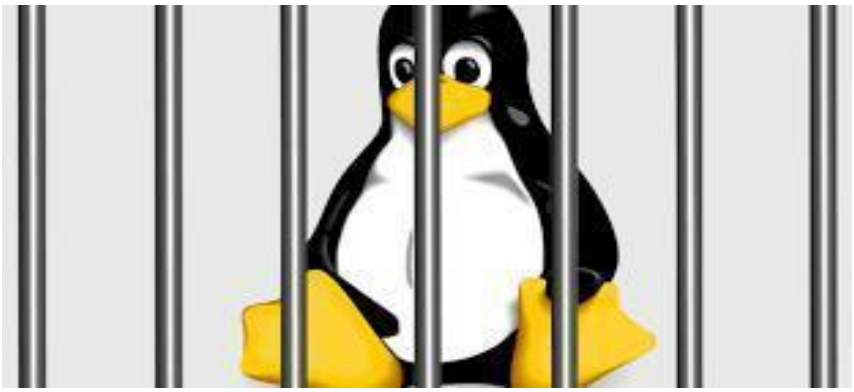
به نام خدا

آزمایشگاه سیستم عامل

پروژه چهارم: همگام سازی

طراحان: امیرمهدی جودی، علی عباسی



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 چیست؟ توابع pushcli و popcli به چه منظور استفاده شده و چه تفاوتی با cli و sti دارند؟

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.

2) مختصری راجع به تعامل میان پردازه‌ها توسط دو تابع مذکور توضیح دهید. چرا در مثال تولیدکننده/مصرف‌کننده[9] استفاده از قفل‌های چرخشی ممکن نیست.

3) حالات مختلف پردازه‌ها در xv6 را توضیح دهید. تابع sched چه وظیفه ای دارد؟

یک مشکل در توابع دسته دوم عدم وجود نگهدارنده[10] قفل است. به این ترتیب حتی پردازهای که قفل را در اختیار ندارد می‌تواند با فراخوانی تابع releasesleep قفل را آزاد نماید.

4) 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.

5) یکی از روش‌های افزایش کارایی در بارهای کاری چندریسه‌ای استفاده از حافظه تراکنشی[11] بوده که در کتاب نیز به آن اشاره شده است. به عنوان مثال این فناوری در پردازنده‌های جدیدتر اینتل[12] تحت عنوان

[8] Busy Waiting
[9] Producer Consumer
[10] Owner
[11] Transcational Memory
[12] Intel

افزونه‌های همگام‌سازی تراکنشی[13] (TSX) پشتیبانی می‌شود.[14] آن را مختصراً شرح داده و نقش حذف قفل[15] را در آن بیان کنید؟

شبیه‌سازی مسئله *readers-priority readers-writers*

در این بخش از پروژه، ابتدا به پیاده‌سازی ساختار هماهنگ‌سازی سمافور[16] در سطح هسته خواهید پرداخت و سپس از آن در شبیه‌سازی مسئله readers-writers استفاده خواهید کرد. برای این منظور از سمافور شمارشی[17] استفاده خواهیم کرد که با استفاده از آن می‌توان اجازه حضور تعداد مشخصی پردازه را به صورت همزمان در ناحیه بحرانی داد و پس از آن تعداد، باقی پردازه‌ها باید پشت سمافور منتظر بمانند. در اینجا سمافور را به صورتی پیاده‌سازی می‌کنیم که در صورتی که پردازه‌ای اجازه ورود به آن را نیافت، به حالت خواب رفته و در صف قرار می‌گیرد. سپس بعد از این که یکی از پردازه‌ها از ناحیه بحرانی خارج شد، پردازه‌ها را به ترتیب زمان ورود از صف خارج کرده و اجازه ورود به ناحیه بحرانی را به آن‌ها می‌دهیم.

sem_init(i, v) system call: سمافور در خانه i ام آرایه را با تعداد v برای حداکثر پردازه‌های درون ناحیه بحرانی ایجاد می‌کند.

sem_acquire(i) system call: زمانی که یک پردازه بخواهد وارد ناحیه بحرانی شود، این فراخوانی سیستمی را صدا می‌زند.

sem_release(i) system call: زمانی که یک پردازه بخواهد از ناحیه بحرانی خارج شود، این فراخوانی سیستمی را صدا می‌زند.

---

[13] Transactional Synchronization Extensions

[14] به علت وجود اشکال‌های امنیتی، در اکثر ریزمعماری‌های کنونی، غیرفعال شده است. اما ظاهراً در آینده همچنان پشتیبانی خواهد شد.

[15] Lock Elision

[16] Semaphore

[17] Counting Semaphore

حال باید مسئله readers-writers با اولویت به reader ها را با ۳ خواننده و ۲ نویسنده شبیه‌سازی کنید. برای این کار، می‌بایست در سطح کاربر برنامه readers-writers را به همراه یک برنامه آزمون بنویسید. برای داده مشترک بین پردازه ها می‌توانید از یک فایل مشترک بین آن ها استفاده کنید متن درون فایل را بخوانید یا درون آن بنویسید. توجه کنید پیاده‌سازی شما **نباید مشکل بن‌بست**[18] **را داشته باشد.** برای شبیه‌سازی این مسئله می‌توانید از روشی که در کتاب توضیح داده شده یا هر روش خلاقانه دیگری استفاده کنید. **توجه** داشته باشید که مسئله readers-writers با اولویت به reader ها یک نوع خاص از مسئله readers-writers است که در آن خواننده‌ها اولویت بالاتری دارند. این اولویت به این معنی است که اگر فایل برای خواندن باز شده باشد، هیچ خواننده ای نباید منتظر بماند.

سایر نکات:

- تمیزی کد و 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.
- فقط فایل‌های تغییر یافته و یا افزوده شده را به صورت ZIP بارگذاری نمایید.
- پاسخ تمامی سوالات را در کوتاه‌ترین اندازه ممکن در گزارش خود بیاورید.
- 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.

[18] Deadlock

- 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

amirmahdiijoudi@gmail.com

موفق باشید