



به نام خدا

آزمایشگاه سیستم عامل

پروژه چهارم: همگام سازی

طراحان: فراز شاهسون، سجاد علی زاده



مقدمه

در این پروژه با سازوکارهای همگام سازی^۱ سیستم عامل ها آشنا خواهید شد. با توجه به این که سیستم عامل xv6 از ریشه های سطح کاربر پشتیبانی نمی کند همگام سازی در سطح پردازنده ها مطرح خواهد بود. همچنین به علت عدم پشتیبانی از حافظه مشترک در این سیستم عامل، همگام سازی در سطح هسته صورت خواهد گرفت. به همین سبب مختصری راجع به این قسم از همگام سازی توضیح داده خواهد شد.

¹ Synchronization Mechanisms

ضرورت همگام سازی در هسته سیستم عامل ها

هسته سیستم عامل ها دارای مسیرهای کنترلی^۲ مختلفی می باشد. به طور کلی، دنباله دستورالعمل های اجرا شده توسط هسته جهت مدیریت فراخوانی سیستمی، وقفه یا استثنا این مسیرها را تشکیل می دهند. در این میان برخی از سیستم عامل ها دارای هسته با ورود مجدد^۳ می باشند. بدین معنی که مسیرهای کنترلی این هسته ها قابلیت اجرای همروند^۴ دارند. تمامی سیستم عامل های مدرن کنونی این قابلیت را دارند. مثلاً 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 می تواند 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 ماهیت های مختلف کدهای اجرایی هسته لحاظ گردد.

هر مسیر کنترلی هسته در یک متن خاص اجرا می گردد. اگر کد هسته به طور مستقیم یا غیرمستقیم توسط برنامه سطح کاربر اجرا گردد، در متن پردازش^۵ اجرا می گردد. در حالی که کدی که در نتیجه وقفه اجرا می گردد در متن وقفه^۶ است. به این ترتیب فراخوانی سیستمی و استثناها در متن پردازش فراخوانده هستند. در حالی که وقفه در متن وقفه اجرا می گردد. به طور کلی در سیستم عامل ها 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

^۲ Control Path

^۳ Reentrant Kernel

^۴ Concurrent

^۵ Process Context

^۶ Interrupt Context

همگام‌سازی آنها نباید منجر به مسدود شدن آنها گردد، مثلاً از قفل‌های چرخشی^۷ استفاده گردد یا در پردازنده‌های تک هسته‌ای وقفه غیر فعال گردد.

همگام‌سازی در xv6

قفل‌گذاری در هسته xv6 توسط دو سری تابع صورت می‌گیرد. دسته اول شامل توابع acquire() (خط ۱۵۷۳) و release() (خط ۱۶۰۱) می‌شود که یک پیاده‌سازی ساده از قفل‌های چرخشی هستند. این قفل‌ها منجر به انتظار مشغول^۸ شده و در حین اجرای ناحیه بحرانی وقفه را نیز غیرفعال می‌کنند. (۱) 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 ۴۶۲۱) و releasesleep() (خط ۴۶۳۳) بوده که مشکل انتظار مشغول را حل نموده و امکان تعامل میان پردازنده‌ها 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.

(۲) مختصری راجع به تعامل میان پردازنده‌ها توسط دو تابع مذکور توضیح دهید. چرا در مثال تولیدکننده/مصرف‌کننده^۹ استفاده از قفل‌های چرخشی ممکن نیست.

(۳) حالات مختلف پردازنده‌ها در xv6 را توضیح دهید. تابع sched() چه وظیفه‌ای دارد؟ یک مشکل در توابع دسته دوم عدم وجود نگهدارنده^{۱۰} قفل است. به این ترتیب حتی پردازنده‌ای که قفل را در اختیار ندارد می‌تواند با فراخوانی تابع releasesleep() قفل را آزاد نماید.

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

⁷ Spin Locks

⁸ Busy Waiting

⁹ Producer Consumer

¹⁰ Owner

(۵) یکی از روش‌های افزایش کارایی در بارهای کاری چندریسه‌ای استفاده از حافظه تراکنشی^{۱۱} بوده که در کتاب نیز به آن اشاره شده است. به عنوان مثال این فناوری در پردازنده‌های جدیدتر اینتل^{۱۲} تحت عنوان افزونه‌های همگام‌سازی تراکنشی^{۱۳} (TSX) پشتیبانی می‌شود.^{۱۴} آن را مختصراً شرح داده و نقش حذف قفل^{۱۵} را در آن بیان کنید؟

شبیه‌سازی مسئله فلاسف خورنده

در این بخش از پروژه، ابتدا به پیاده‌سازی ساختار هماهنگ‌سازی سمافور^{۱۶} در سطح هسته خواهید پرداخت و سپس از آن در شبیه‌سازی اجرای مسئله فلاسف خورنده^{۱۷} استفاده خواهید کرد. برای این منظور از سمافور شمارشی^{۱۸} استفاده خواهیم کرد که با استفاده از آن می‌توان اجازه حضور تعداد مشخصی پردازنده را به صورت همزمان در ناحیه بحرانی داد و پس از آن تعداد، باقی پردازنده‌ها باید پشت سمافور منتظر بمانند. در اینجا سمافور را به صورتی پیاده‌سازی می‌کنیم که در صورتی که پردازنده‌ای اجازه ورود به آن را نیافت، به حالت خواب رفته و در صف قرار می‌گیرد. سپس بعد از این که یکی از پردازنده‌ها از ناحیه بحرانی خارج شد، پردازنده‌ها را به ترتیب زمان ورود از صف خارج کرده و اجازه ورود به ناحیه بحرانی را به آن‌ها می‌دهیم. ابتدا یک آرایه پنج تایی از سمافور در سطح سیستم ایجاد کنید که برنامه‌های سطح کاربر، از طریق فراخوانی‌های سیستمی زیر می‌توانند به آن‌ها دسترسی داشته باشند.

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

¹¹ Transactional Memory

¹² Intel

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 Dining Philosophers

¹⁸ Counting Semaphore

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

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

حال باید مسئله فلاسفه خورنده را با پنج فیلسوف شبیه‌سازی کنید. برای این کار، می‌بایست در سطح کاربر برنامه فیلسوف‌ها را به همراه یک برنامه آزمون بنویسید. برای هر قاشق، از یک متغیر صحیح^{۱۹} استفاده کنید که شماره فیلسوفی که در لحظه آن را در اختیار دارد را داشته باشد و اگر فیلسوفی آن را در اختیار ندارد، مقدارش برابر ۱- باشد. توجه کنید پیاده‌سازی شما **نباید مشکل بن‌بست**^{۲۰} را داشته باشد. برای شبیه‌سازی این مسئله می‌توانید از روشی که در کتاب توضیح داده شده (پیاده‌سازی متغیر شرط با استفاده از سمافور و سپس استفاده از مانیتور) یا هر روش خلاقانه دیگری استفاده کنید.

¹⁹ Integer

²⁰ Deadlock

سایر نکات:

- تمیزی کد و مدیریت حافظه مناسب در پروژه از نکات مهم پیاده‌سازی است.
- از لاگ‌های مناسب در پیاده‌سازی 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 لزوماً یکسان نخواهد بود.
- در صورت تشخیص تقلب، نمره هر دو گروه صفر در نظر گرفته خواهد شد.
- فصل ۴ و انتهای فصل ۵ کتاب xv6 می‌تواند مفید باشد.
- هرگونه سوال در مورد پروژه را از طریق ایمیل‌های طراحان می‌توانید مطرح نمایید.

sajjadalizadeh2000@gmail.com

faraz.shahsavan@ut.ac.ir

موفق باشید