بسم الله الرحمن الرحیم

پروژه چهارم درس آزمایشگاه سیستم عامل

دکتر کارگهی

محمد امین توانایی - ۸۱۰۱۰۱۳۹۶

سید علی تهامی - ۸۱۰۱۰۱۳۹۷

مهدی وجهی - ۸۱۰۱۰۱۵۵۸

سوال ۱:

به طور کلی، interrupt ها در طول اجرای یک critical section غیرفعال می‌شوند تا از race condition ها جلوگیری شود و یکپارچگی داده‌ها تضمین گردد. دلایل دقیق‌تر این امر به شرح زیر است:

I. جلوگیری از Context Switch ها: غیرفعال کردن interrupt ها از context switch به task دیگری که ممکن است به همان داده دسترسی داشته باشد جلوگیری می‌کند. این مهم است زیرا اگر interrupt ای در حین اجرای یک process در critical section رخ دهد، interrupt handler می‌تواند به همان داده دسترسی پیدا کند و منجر به ناسازگاری‌ها شود.

II. اجتناب از Deadlock ها: اگر interrupt ای در حین اجرای critical section رخ دهد و interrupt handler سعی کند همان lock را به دست آورد، می‌تواند منجر به وضعیت deadlock شود. با غیرفعال کردن interrupt ها، اطمینان حاصل می‌کنیم که lock به درستی آزاد می‌شود قبل از اینکه interrupt handler بتواند آن را به دست آورد.

III. تضمین Atomicity: Critical section ها اغلب شامل به‌روزرسانی shared resource ها هستند. برای اطمینان از اینکه این عملیات‌ها atomic هستند (یعنی کاملاً انجام می‌شوند یا اصلاً انجام نمی‌شوند)، interrupt ها غیرفعال می‌شوند. این امر از ایجاد وضعیت نیمه‌کاره عملیات توسط یک interrupt جلوگیری می‌کند.

IV. حفظ System Responsiveness: سیستم‌های real-time باید تضمین کنند که چه مدت زمانی برای پاسخ به interrupt ها طول می‌کشد، اما critical section ها می‌توانند به طور دلخواه طولانی باشند. بنابراین، interrupt ها برای کوتاه‌ترین زمان ممکن خاموش می‌مانند تا system responsiveness حفظ شود.

سوال ۲:

enum procstate { UNUSED, EMBRYO, SLEEPING, RUNNABLE, RUNNING, ZOMBIE }

این enum حالت‌های ممکن یک فرآیند را تعریف می‌کند:

حالت‌های فرآیند

- UNUSED:

نشان می‌دهد که ورودی جدول فرآیند در حال حاضر توسط هیچ منبعی استفاده نمی‌شود.

- EMBRYO:

نشان‌دهنده فرآیندی است که در حال ایجاد است. در این حالت، منابعی مانند حافظه و توصیف‌کننده فایل به فرآیند اختصاص داده می‌شود.

- SLEEPING:

نشان می‌دهد فرآیند منتظر در دسترس قرار گرفتن یک منبع یا رخ دادن یک رویداد است. فرآیند در حالت SLEEPING برای اجرا زمان‌بندی نمی‌شود.

- RUNNABLE:

نشان می‌دهد فرآیند آماده اجراست و منتظر تخصیص به CPU است. فرآیند تمام منابع مورد نیاز برای اجرا را دارد و فقط منتظر زمان CPU از زمان‌بندی‌کننده است.

- RUNNING:

نشان می‌دهد فرآیند در حال حاضر روی یک CPU در حال اجراست. حداکثر یک فرآیند RUNNING به ازای هر CPU می‌تواند وجود داشته باشد.

- ZOMBIE:

نشان می‌دهد فرآیند اجرایش را به پایان رسانده (خارج شده)، اما فرآیند والد هنوز وضعیت خروج آن را جمع‌آوری نکرده است.

انتقال به حالت RUNNABLE

یک فرآیند می‌تواند به چندین روش به حالت RUNNABLE منتقل شود:

1. تازه ایجاد شده
2. از خواب برخاسته
3. پیش‌گیری شده
4. تکمیل I/O
5. آزاد شدن قفل
6. مدیریت سیگنال

تابع sched:

تابع sched مسئول زمان‌بندی فرآیندها در سیستم عامل است. این بخشی از زمان‌بندی‌کننده فرآیند است که تصمیم می‌گیرد کدام فرآیند بعدی اجرا شود. به این صورت کار می‌کند:

1. تعویض زمینه:

xv6 دو نوع تعویض زمینه انجام می‌دهد: از thread هسته یک فرآیند به thread زمان‌بندی‌کننده CPU فعلی، و از thread زمان‌بندی‌کننده به thread هسته یک فرآیند.

2. حالت‌های فرآیند:

وظیفه زمان‌بندی‌کننده انتخاب یک فرآیند از حالت RUNNABLE و تغییر آن به حالت RUNNING است.

3. مکانیزم خواب و بیداری:

xv6 توابع sleep و wakeup را ارائه می‌دهد که معادل توابع wait و signal یک متغیر شرطی هستند.

4. چندوظیفگی:

اگر دو فرآیند بخواهند روی یک CPU اجرا شوند، xv6 آنها را چندوظیفه می‌کند و چندین بار در ثانیه بین اجرای یکی و دیگری تغییر می‌کند.

سوال ۳:

روش MSI (Modified-Shared-Invalid) یکی از پروتکل‌های مدیریت همگام‌سازی حافظه پنهان (Cache Coherence) در سیستم‌های چندپردازنده‌ای است. این پروتکل سه حالت برای داده‌های ذخیره‌شده در حافظه پنهان تعریف می‌کند:

1. **Modified (تغییریافته):** داده در کش وجود دارد و از حافظه اصلی متفاوت است (تغییر داده شده است). در این حالت، داده فقط در کش موجود بوده و در حافظه اصلی نوشته نشده است.
2. **Shared (اشتراکی):** داده در حافظه پنهان موجود است و نسخه‌ای از آن در حافظه اصلی یا کش‌های دیگر نیز وجود دارد، اما بدون تغییر.
3. **Invalid (نامعتبر):** داده در کش دیگر معتبر نیست و نمی‌توان از آن استفاده کرد.

این پروتکل تضمین می‌کند که داده‌ها در حافظه‌های پنهان مختلف همگام باقی بمانند و تغییرات به‌صورت صحیح به پردازنده‌های دیگر اعلام شود.

سوال ۴:

Ticket lock یک مکانیسم همگام‌سازی یا الگوریتم قفل‌گذاری است که نوعی spinlock است و از "ticket" ها برای کنترل اینکه کدام thread اجازه ورود به critical section را دارد، استفاده می‌کند. مفهوم اساسی ticket lock مشابه سیستم مدیریت صف بلیطی است که در نانوایی‌ها و فروشگاه‌ها برای خدمت‌رسانی به مشتریان به ترتیب ورودشان استفاده می‌شود.

دو مقدار صحیح وجود دارد که از 0 شروع می‌شوند. اولین مقدار queue ticket و دومی dequeue ticket است. وقتی یک thread می‌رسد، به صورت atomic یک ticket دریافت کرده و سپس آن را افزایش می‌دهد. سپس مقدار ticket خود را، قبل از افزایش، با مقدار dequeue ticket مقایسه می‌کند. اگر یکسان باشند، به thread اجازه ورود به critical section داده می‌شود. اگر یکسان نباشند، پس thread دیگری باید در حال حاضر در critical section باشد و این thread باید در حالت busy-wait قرار گیرد یا yield شود.

از دیدگاه ticket lock ،cache coherence ها را می‌توان به شرح زیر تحلیل کرد:

- Cache coherence اطمینان می‌دهد که به‌روزرسانی قفل توسط پردازنده‌های دیگر دیده می‌شود. فرآیندی که قفل را در حالت انحصاری به دست می‌آورد، اولین به‌روزرسانی قفل را انجام می‌دهد.

- قفل می‌تواند cached شود، و cache coherence اطمینان می‌دهد که به‌روزرسانی قفل توسط پردازنده‌های دیگر دیده می‌شود. فرآیندی که قفل را در حالت انحصاری به دست می‌آورد، اولین به‌روزرسانی قفل را انجام می‌دهد.

- Ticket lock در واقع منصفانه است، اما عملکرد آن تقریباً برابر با الگوریتم pthread spinlock است. معرفی ticket lock عمدتاً به دلایل انصاف است.

سوال 5

پیچیدگی در عیب یابی

احتمال سو استفاده

سوال 6

میتوان دو حالت برای این قفل متصور بود

خواندن

در این حالت چند ریسه میتوانند با هم بخوانند مشروط بر اینکه کسی ننویسد.

نوشتن

اگر ریسه ای دارد مینویسد در این صورت هیچ ریسه ی دیگری نمیتواند قفل را بگیرد.

برتری ها

در جاهایی که تعداد زیاد خواندن داریم بهتر عمل میکند پرفورمنس بیشتری دارد.

توانایی موازی سازی بیشتری دارد.