آزمایشگاه سیستم‌های عامل

آزمایش شماره‌ی سه: همگام‌سازی

تاریخ تحویل: 7 آذر 1395

اهداف آزمایش

- آشنایی با همگام‌سازی در لینوکس
- آشنایی با نمونه‌ای از پیاده‌سازی مکانیسم‌های همگام‌سازی
- پیاده‌سازی و اشکال‌زدایی یک مکانیسم جدید در هسته‌ی لینوکس

چکیده

در این آزمایش ابتدا به بررسی نحوه‌ی پیاده‌سازی یکی از مکانیسم‌های همگام‌سازی در لینوکس خواهیم پرداخت. در ادامه با استفاده از دانش به‌دست‌آمده از بخش قبل، به پیاده‌سازی یک مکانیسم جدید همگام‌سازی خواهیم پرداخت و در انتها با طراحی یک سناریو، درستی عملکرد مکانیسم جدید پیاده‌سازی شده را نشان خواهیم داد.

شرح آزمایش

این آزمایش شامل دو بخش است. در بخش اول به بررسی کد منبع مکانیسم همگام‌سازی سمافور در هسته‌ی لینوکس خواهید پرداخت. در بخش بعدی یک مکانیسم جدید همگام‌سازی توصیف‌شده که آن را پیاده‌سازی خواهید کرد.

آشنایی با پیاده‌سازی سمافور در لینوکس

بخش عمده‌ای از تعریف‌ها و پیاده‌سازی‌های مرتبط با سمافور در فایل‌های semaphore.h[1] و semaphore.c[2] آمده است. برای آشنایی بیشتر با مقدمات پیاده‌سازی یک مکانیسم همگام‌سازی، به سؤالاتی از این فایل‌ها خواهیم پرداخت:

1. ساختار داده‌ی اصلی سمافور با نام semaphore در فایل semaphore.h تعریف شده است. به‌طور کامل توضیح دهید[3] که هریک از 3 عضو این ساختار چه کاربردی دارند؟
2. توابع و ماکروهایی که برای مقداردهی اولیه به سمافور تعریف‌شده‌اند را نام برده و به‌طور مختصر شرح دهید.
3. به پیاده‌سازی توابع ()up و ()down در فایل semaphore.c مراجعه کنید. توضیح دهید که برای حفاظت از ناحیه‌ی بحرانی در این دو تابع چگونه عمل شده است. توابع داخلی را نیز بررسی کنید. مثل down_common و ...
4. به‌طور کامل توضیح دهید که تابع ()up در ناحیه‌ی بحرانی خود چه می‌کند؟
5. به‌طور کامل توضیح دهید که تابع ()down در ناحیه‌ی بحرانی خود چه می‌کند؟

در هنگام تحویل باید به تمام روند کار سمافور مسلط باشید.

---

[1] در پوشه‌ی include/linux/

[2] در پوشه‌ی kernel/

[3] منظور از توضیح کامل، توضیحی است که گویای تمام مفاهیم بکار رفته در پیاده‌سازی باشد. برای مثال ممکن است لازم باشد که یک ساختار، تابع یا عضو پراهمیت، به‌طور جزئی شرح داده‌شود. در غالب این موارد، توضیحی فراتر از کامنت‌های موجود در کد منبع لازم است.

پیاده سازی مکانیسم جدید همگام‌سازی

حال میخواهیم سمافور جدیدی پیاده سازی کنیم که تعدادی از مشکلات فعلی را رفع کند.

در ابتدا میخواهیم پراسس بعدی جهت ورود به سمافور را از بین پراسس های منتظر انتخاب کنیم. برای این کار به اولویت پراسس ها توجه میکنیم.

برای این کار جستجو کنید که اولویت پراسس ها چگونه در سیستم عامل نگه داری میشوند.

حال با درگیر شدن اولویت در انتخاب پراسس، دو مشکل priority inversion و starvation رخ میدهد که باید آن ها را حل کنید.

پیاده‌سازی شما باید نوع[4] newsem را به همراه 3 فراخوان سیستمی زیر را در اختیار کاربر قرار دهد:

int newsem_init(newsem* instance, int n): اشاره‌گر به یک newsem را دریافت کرده و مقداردهی اولیه‌ی آن را انجام می‌دهد. دقت کنید که سمافور اجازه ی ورود n پراسس به داخل را میدهد.

- int newsem_wait(newsem* instance): مقدار سمافور را کاهش می‌دهد و اگر مقدار منفی شود وظیفه‌ی درخواست دهنده را بلوکه می‌کند.
- int newsem_signal(newsem* instance): مقدار سمافور را افزایش می‌دهد و در صورت وجود پردازه‌ی منتظر، پر اولویت‌ترین آن‌ها (طبق توضیح قبل) را از حالت بلوکه خارج می‌کند.

مقدار بازگردانده‌شده توسط توابع بالا باید نشان‌دهنده‌ی موفقیت فراخوانی و یا کد خطایی که رخ‌داده باشد. تخصیص مقادیر کدها در این پروژه دلخواه است. البته معمولاً این کدها در سیستم‌های عامل مختلف دارای مقادیر استاندارد هستند. برای مثال غالباً مقدار صفر نشان‌دهنده‌ی موفقیت است و مقدار منفی عدم موفقیت را می‌رساند. ممکن است متغیر errno نیز در مواردی جزئیات نوع خطا را نشان دهد. در این پروژه از شما انتظار می‌رود که نوع خطای رخ‌داده را نیز بازگردانید.

تعریف newsem و فراخوانی توابع باید به‌صورت زیر قابل‌استفاده باشد:

```
newsem * ns1 = new newsem;
```

[4] type

```
int ret_val;
ret_val = newsem_init( ns1, 4);
ret_val = newsem_wait( ns1);
ret_val = newsem_signal( ns1);
```

در انتخاب ساختمان داده و الگوریتم مورد استفاده ی خود، دقت کنید و آنرا بهینه انتخاب کنید و تحلیل مناسبی برای انتخاب و پیاده سازیتان داشته باشید.

طراحی سناریوی تست

برای اطمینان از عملکرد درست پیاده‌سازی شما، نیاز به یک برنامه‌ی C است که اجرای آن نشان‌دهنده‌ی این مطلب باشد. این برنامه باید درستی عملکرد به‌عنوان یک مکانیسم همگام‌سازی و همین‌طور درستی پیاده سازی عملکرد خواسته شده را نیز نشان دهد. یک فایل نمونه برای کمک به شما برای طراحی تست هایتان آپلود میشود.

سایر نکات

- پاسخ به سؤالات را در گزارش خود بیاورید. گزارش باید به‌صورت تایپ‌شده یا تصویر خوانا آپلود شود.
- تنها بخش‌های تغییریافته‌ی هسته و کدهای مربوط به سناریوی تست خود را به همراه گزارش آپلود کنید (کل هسته را آپلود نکنید).
- طبیعت پروژه‌های آزمایشگاه به‌گونه‌ای است که ممکن است با مشکلات پیش‌بینی‌نشده مواجه شوید. دستیاران آموزشی در رفع این مشکلات به شما کمک خواهند کرد، ولی مسئولیت انجام درست پروژه به عهده‌ی خود شما است. بنابراین توصیه می‌شود که پروژه را زود شروع کنید.
- پروژه‌های آزمایشگاه باید در گروه‌های سه‌نفره انجام شوند. همه‌ی اعضای گروه باید روی همه‌ی قسمت-های پروژه تسلط داشته باشند و هریک از اعضا متناسب با میزان تسلط نمره دهی خواهد شد.
- سؤالات خود را در فروم مخصوص پروژه در CECM مطرح کنید و در صورت امکان به سؤالات دیگران پاسخ دهید. دستیاران آموزشی درس نیز به پرسش‌های مطرح‌شده در سایت پاسخ خواهند داد.