



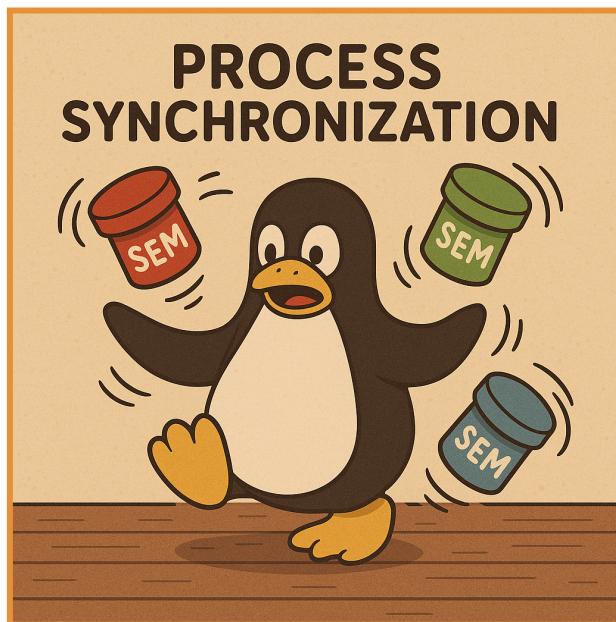
به نام خدا



آزمایشگاه سیستم عامل - بهار ۱۴۰۴

پروژه‌ی چهارم: همگام‌سازی

طراحان: مهندی حاجی - علیرضا حسینی



## مقدمه

در این پروژه با سازوکارهای همگام‌سازی<sup>1</sup> سیستم عامل‌ها آشنا خواهید شد. با توجه به این موضوع که سیستم عامل xv6 از ریسه<sup>2</sup>های 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 مختصراً راجع به این قسم از همگام‌سازی توضیح داده خواهد شد.

<sup>1</sup> Synchronization

<sup>2</sup> Theard

## ضرورت همگام سازی در هسته سیستم عامل ها

هسته سیستم عامل ها دارای مسیرهای کنترلی<sup>۳</sup> مختلفی می باشد . به طور کلی ، دنباله دستورالعمل های اجرا شده توسط هسته جهت مدیریت فرآخوانی سیستمی<sup>۴</sup>، وقفه یا استثنا این مسیرها را تشکیل میدهند. در این میان برخی از سیستم عامل ها دارای هسته با ورود مجدد<sup>۵</sup> می باشند ; بدین معنی که مسیرهای کنترلی این هسته ها قابلیت اجرای همرونده<sup>۶</sup> دارند. تمامی سیستم عامل های مدرن کنونی این قابلیت را دارند. مثلاً ممکن است برنامه سطح کاربر در میانه اجرای فرآخوانی سیستمی در هسته باشد که وقفه ای رخ دهد. به این ترتیب در حین اجرای یک مسیر کنترلی در هسته (اجرای کد فرآخوانی 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 همگامسازی باید ماهیتهای مختلف کدهای اجرایی هسته لحاظ گردد.

یک مثال معروفی که برای توضیح همگام سازی و قفل گذاری استفاده می شود به صورت زیر است:

به عنوان مثال از قفل گذاری، پلی را تصور کنید که دارای محدودیت وزنی بر روی خود می باشد. به طوری که در هر لحظه تنها یک خودرو میتواند از روی پل عبور کند و در غیر این صورت فرو میریزد. قفل همانند یک نگهبان در ورودی پل مراقبت می کند که تنها زمانی به خودرو جدید اجازه ورود بدهد که هیچ خودرویی بر روی پل نباشد.

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<sup>۷</sup> اجرا می گردد. در حالی که کدی که در نتیجه وقفه اجرا می گردد در متن وقفه<sup>۸</sup> است.

به این ترتیب فرآخوانی سیستمی و استثنایها در متن پردازه فرآخواننده هستند. در حالی که وقفه در متن وقفه اجرا میگردد. 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 همگامسازی آنها نباید منجر به مسدود شدن آنها گردد. مثلاً از قفلهای چرخشی<sup>۹</sup> استفاده گردد یا در پردازنده های تک هسته ای وقفه غیرفعال گردد.

<sup>3</sup> Control Flow (Control Path)

<sup>4</sup> System call

<sup>5</sup> Reentrant Kernel

<sup>6</sup> Concurrent

<sup>7</sup> Process Context

<sup>8</sup> Interrupt Context

<sup>9</sup> Spinlock

## چگونگی همگام سازی در سیستم عامل XV6

قفل گذاری در هسته xv6 توسط دو سری تابع صورت میگیرد. دسته اول شامل توابع release و acquire میشود که یک پیاده سازی ساده از قفلهای چرخشی هستند. این قفلها منجر به waiting شده و در حین اجرای ناحیه بحرانی وقفه را نیز غیرفعال میکنند.

- **علت غیرفعال کردن وقفه چیست؟ توابع popcli و pushcli به چه منظور استفاده شده و چه تفاوتی با cli و sti دارند؟**

دسته دوم شامل توابع acquiresleep و releasesleep میباشد که مشکل انتظار مشغول را حل نموده و امکان تعامل میان پردازه ها 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 پردازندۀ را رها میکنند.

- **حالات مختلف پردازه ها در xv6 را توضیح دهید. تابع sched چه وظیفه ای دارد؟**

## مسئله آرایشگر بی حال - Sleeping Barber



مسئله‌ی آرایشگر خفته<sup>۱۰</sup> یکی از مسائل کلاسیک در حوزه‌ی همزمانی<sup>۱۱</sup> در سیستم‌عامل‌هاست که برای مدیریت دسترسی چند نخ<sup>۱۲</sup> به منابع مشترک استفاده می‌شود. این مسئله برای اولین بار توسط Dijkstra مطرح شد و هدف آن مدل‌سازی صحیح یک سناریوی واقعی از طریق استفاده از semaphores یا دیگر مکانیزم‌های همزمانی است.

سناریو مسئله به شکل زیر است :

فرض کن یک آرایشگاه داریم با ویژگی‌های زیر:

- یک آرایشگر
- یک صندلی آرایش برای اصلاح مشتری
- یک اتاق انتظار با تعداد مشخص صندلی (در مسئله ما ۵ صندلی)
- وقتی یک مشتری وارد می‌شود :
  - اگر آرایشگر آزاد باشد، مستقیم اصلاح می‌شود.
  - اگر آرایشگر مشغول است و صندلی خالی در اتاق انتظار وجود دارد، آنجا می‌نشیند تا نوبتش شود.
  - اگر هیچ صندلی خالی نیست، مشتری می‌رود.

<sup>10</sup> Sleeping Barber

<sup>11</sup> synchronization

<sup>12</sup> thread

- اگر هیچ مشتری‌ای نباشد، آرایشگر می‌خوابد (sleep).
  - زمانی که مشتری وارد می‌شود و آرایشگر خواب است، مشتری او را بیدار می‌کند.
- برای اطلاعات بیشتر میتوانید به رفرنس‌های ۴ و ۵ و ۶ مراجعه کنید.**

در این بخش از پرژوهه، ابتدا به پیاده سازی ساختار هماهنگ سازی سمافور در سطح هسته خواهید پرداخت و سپس از آن در شبیه سازی اجرای مسئله آرایشگر بی‌حال استفاده خواهید کرد. برای این مثلاً منظور میتواند از سمافور شمارشی<sup>۱۳</sup> استفاده کنید 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 یا به کل قید دسترسی به منبع مشترک را بزنند.

ابتدا یک آرایه شش تابی (شامل پنج مشتری و یک آرایشگر) از سمافور در سطح سیستم ایجاد کنید که برنامه‌های سطح کاربر، از طریق فراخوانی‌های سیستمی زیر میتوانند به آنها دسترسی داشته باشند.

وقتی آرایشگر آماده اصلاح نیست و باید منتظر مشتری باشد: barber\_sleep()

وقتی یک مشتری وارد آرایشگاه می‌شود: customer\_arrive()

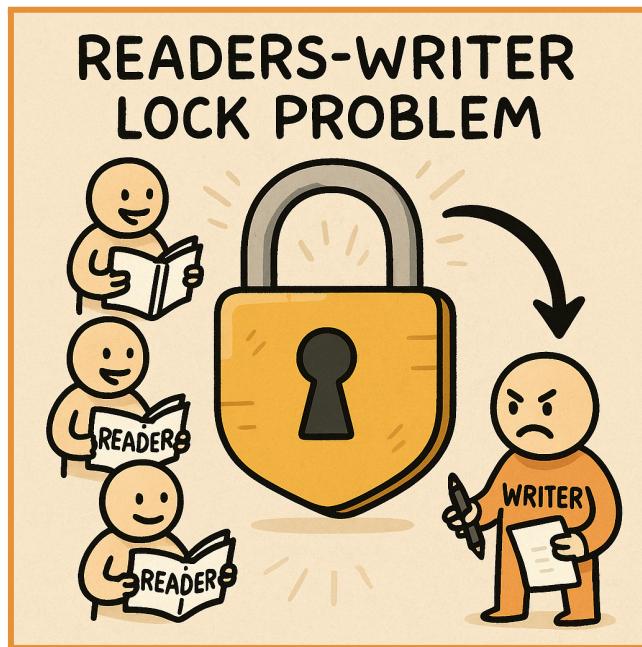
انجام عملیات اصلاح: cut\_hair()

منطق ورود مشتری‌ها و بیدار کردن آرایشگر باید در توابع barber\_sleep و customer\_arrive و پیاده‌سازی شود، به‌گونه‌ای که شرایط بحرانی و ناسازگاری رخ ندهد. از قفل‌های چرخشی یا قفل‌های خواب استفاده کنید تا از رقابت میان ریسه‌ها جلوگیری شود.

حال باید مسئله آرایشگر بی‌حال را با پنج صندلی شبیه سازی کنید. برای این کار، می‌بایست در سطح کاربر برنامه آرایشگر بی‌حال را به همراه یک برنامه آزمون بنویسید. مطمئن شوید مشتری‌هایی که باید منتظر بمانند، درست در صفت می‌مانند و بقیه رد می‌شوند.

<sup>13</sup> Counting Semaphore

## 14 قفل Readers-Writers



مسئله قفل خوانندگان-نویسندهان<sup>۱۵</sup> یکی از مسائل کلاسیک در همروندي<sup>۱۶</sup> و برنامه‌نویسی همزمان<sup>۱۷</sup> است. این مسئله زمانی مطرح می‌شود که چند خواننده (reader) و نویسنده (writer) بخواهند به یک منبع مشترک دسترسی داشته باشند.

- خواننده‌ها فقط داده را می‌خوانند و آن را تغییر نمی‌دهند.
- نویسنده‌ها داده را تغییر می‌دهند (می‌نویسنند).

ما می‌خواهیم دسترسی به منبع طوری مدیریت شود که:

1. چند خواننده می‌توانند به طور همزمان بخوانند (چون تغییری ایجاد نمی‌کنند، تداخلی نیست).
2. اما وقتی نویسنده‌ای در حال نوشتن است، نه خواننده‌ای و نه نویسنده‌ی دیگری نباید به منبع دسترسی داشته باشد.
3. باید بین خوانندگان و نویسندهان تعادل یا اولویت مشخصی برقرار شود، تا نه خواننده‌ها گرسنه<sup>۱۸</sup> بمانند، نه نویسنده‌ها.

<sup>14</sup> Reader-Writer lock

<sup>15</sup> Readers-Writers Lock/Problem

<sup>16</sup> concurrency

<sup>17</sup> multithreading

<sup>18</sup> starvation

قفل خوانندگان-نویسندها را برای حالت تقدیم نویسندها و با قابلیت به خواب رفتن در حین انتظار پیاده سازی نمایید.

برای این قفل شما موظف به پیاده سازی دو فراخوانی سیستمی<sup>19</sup> هستید.

### 1. فراخوانی سیستمی `init_rw_lock`

تابع `init_rw_lock` برای مقداردهی اولیه قفل نویسندها-خوانندگان در هسته سیستم عامل `xv6` استفاده می‌شود.

- ساختارهای لازم مانند سمافورها (semaphores)، متغیرهای شمارنده و قفل‌ها را تنظیم و مقداردهی اولیه می‌کند.

- این تابع معمولاً یک بار و پیش از اجرای فراخوانی‌های `get_rw_pattern` فراخوانی می‌شود.
- هدف آن فراهم کردن امکان همزمانی بین پردازه‌هایی است که به صورت خواننده یا نویسنده قصد دسترسی به داده‌ی مشترک را دارند.

### 2. فراخوانی سیستمی `get_rw_pattern`

تابع `get_rw_pattern` برای تست عملکرد قفل خوانندگان-نویسندها بر اساس یک الگوی دودویی (bit pattern) طراحی شده است. که این bit pattern به عنوان ورودی فراخوانی سیستمی به صورت `int` داده می‌شود ( واضح است که حتماً اولین بیت این الگو همواره یک خواهد بود)

- ورودی این تابع یک عدد صحیح به عنوان الگو (pattern) است.
- این عدد به صورت دودویی تفسیر می‌شود و هر بیت آن نشان‌دهنده‌ی یک عملیات خواندن یا نوشتن است:
  - بیت 0 ⇒ عملیات خواندن
  - بیت 1 ⇒ عملیات نوشتن
- ترتیب عملیات‌ها از سمت چپ به راست در نظر گرفته می‌شود.

<sup>19</sup> System call

**مثال:**

اگر الگوی ورودی ۱۹ باشد، نمایش دودویی آن برابر با ۱۰۰۱۱ است، که به معنی:

- عملیات اول: خواندن
- عملیات دوم: خواندن
- عملیات سوم: نوشتن
- عملیات چهارم: نوشتن
- عملیات پنجم: انجام نمی‌شود (چون فقط ۴ بیت معتبر داریم و پر ارزش ترین بیت همواره یک است)

هر پردازه طبق این الگو عمل می‌کند و با استفاده از قفل پیاده‌سازی شده، به داده‌ی مشترک دسترسی پیدا می‌کند.

در اینجا نیز میتوان فرض نمود که متغیر مشترک یک عدد بوده که با هر بار نوشتن یک واحد به مقدار آن افزوده می‌شود. توابع مربوط به دریافت و رهاسازی هر نوع قفل نیز در هسته پیاده‌سازی می‌گردد. مواردی از قبیل افزایش متناسب مقدار متغیر مشترک و امکان دسترسی همزمان چندین خواننده به متغیر و تقدم خوانندگان بر نویسنده‌گان نمایش داده شود.

- در چه مواردی این قفل نسبت به قفل بلیط مزیت دارد. (میتوانید از رفرنس‌های شماره ۱ و ۲ و ۳ استفاده نمایید)

## منابعی که می‌توانند مفید باشند

1. [https://en.wikipedia.org/wiki/Readers%20writer\\_lock](https://en.wikipedia.org/wiki/Readers%20writer_lock)
2. [https://en.wikipedia.org/wiki/Readers%20writers\\_problem](https://en.wikipedia.org/wiki/Readers%20writers_problem)
3. [https://medium.com/@tusharmalhotra\\_81114/efficiently-managing-concurrency-implementing-a-reader-writer-lock-in-c-95c0c1c709cd](https://medium.com/@tusharmalhotra_81114/efficiently-managing-concurrency-implementing-a-reader-writer-lock-in-c-95c0c1c709cd)
4. [https://en.wikipedia.org/wiki/Sleeping\\_barber\\_problem](https://en.wikipedia.org/wiki/Sleeping_barber_problem)
5. <https://stackoverflow.com/questions/19692515/sleeping-barber-algorithm-with-multiple-barbers>
6. <https://www.baeldung.com/cs/sleeping-barber-problem>

## سایر نکات

- پروژه خود را در Gitlab یا Github پیش برد و در نهایت یک نفر از اعضای گروه کدها را به همراه OS-Lab4-<SID1>-<SID2>-<SID3>.zip پوشش git. و گزارش زیپ کرده و در سامانه با فرمت آپلود نمایید.
- رعایت نکردن مورد فوق کسر نمره را به همراه خواهد داشت.
- بخش خوبی از نمره شما را پاسخ دهی به سوالات مطرح شده که با رنگ قرمز مشخص شده اند، تشکیل میدهد که به شما در درک نحوه کارکرد xv6 و پیاده سازی قسمت سوالات کمک میکند.
- پاسخ سوالها را تا حد ممکن کوتاه بنویسید.
- همه افراد می بایست به پروژه مسلط باشند و نمره تمامی اعضای گروه لزوماً یکسان نیست.
- تمامی مواردی که در جلسه توجیهی، گروه اسکایپ و فروم درس مطرح می شوند، جزئی از پروژه خواهند بود. در صورت وجود هرگونه سوال یا ابهام میتوانید با ایمیل دستیاران مربوطه یا گروه اسکایپی درس در ارتباط باشید.
- این تمرین صرفا برای یادگیری شما طرح شده است. در صورت محرز شدن تقلب در تمرین، مطابق با قوانین درس برخورد خواهد شد.

## موفق باشید