آزمایشگاه سیستم عامل

دستورکار ۱۳: همگام‌سازی فرآیندها

پایه اصلی چندبرنامگی در یک سیستم عامل، قابلیت اجرای چندین برنامه با یکدیگر است. اجرای چند برنامه با یکدیگر در یک محیط تک‌پردازنده بدین مفهوم است که هرچند در هر زمان یک برنامه در حال اجراست ولی چندین برنامه در سیستم عامل هستند که اجرای آنها شروع شده ولی پایان نیافته‌اند. عموماً در محیط‌های چندبرنامه‌ای تعدادی از برنامه‌ها با یکدیگر دارای نواحی داده‌ای مشترک یا حافظه مشترک می باشند. برای استفاده صحیح از این داده‌ها بدون بروز خطا، باید هماهنگی بین آن‌ها وجود داشته باشد. یک فرآیند همکار فرآیندی است که می‌تواند به سایر پردازش‌های در حال اجرا اثر گذارد و یا از آنها تاثیر بگیرد. فرآیندهای همکار ممکن است یا مستقیماً یک فضای آدرس را به اشتراک گذارد و یا اجازه داشته باشد داده‌های یکدیگر را تغییر دهند. به وضعیتی که دو یا چند فرآیند به متغیرهای اشتراکی دسترسی دارند و در نتیجه فرآیندها بستگی به زمان فرآیندها دارد شرایط مسابقه‌ای یا Race condition گویند.

ناحیه بحرانی (critical section) : سیستمی متشکل از n فرآیند که هر فرآیند دارای ناحیه ای در کد است که در آن ناحیه ممکن است متغیرهای مشترک تغییر نمایند یا جدولی به‌روز شود و یا فایلی به اشتراک گذاشته شود. به این بخش از کد ناحیه بحرانی می‌گویند.

برای برطرف کردن شرایط رقابتی راه‌حل مختلفی وجود دارد. هر راه‌حل باید بتواند سه شرایط زیر را ارضا کند:

۱- Mutual exclusion (انحصار متقابل): اگر فرآیندی در حال اجرا در ناحیه بحرانی باشد در آن صورت نمی‌تواند اجرا شود. (هیچ دو فرآیندی نباید هم‌زمان در ناحیه بحرانی باشند.)

۲- Progress (پیشرفت): اگر تعدادی از فرآیندها در ناحیه بحرانی نباشد، فقط آن فرآیندهایی که به ناحیه بحرانی نرسیده‌اند می‌توانند در تصمیم‌گیری درباره اینکه کدام فرآیند وارد ناحیه بحرانی شود شرکت کنند. به عبارت دیگر هیچ فرآیندی در بیرون از ناحیه بحرانی خود نمی‌توانند فرآیندهای دیگر را بلوکه کنند.

۳- Bounded waiting (انتظار محدود): یک فرآیند نباید به طور نامحدود برای ورود به ناحیه بحرانی منتظر باقی بماند.

بن بست (Dead lock): در یک محیط چند برنامه‌ای این امکان وجود دارد که فرآیندهای متفاوتی برای بدست آوردن تعداد محدودی منبع با یکدیگر رقابت کنند. یک فرآیند منابعی را درخواست می‌کند و در صورتی که این منابع در آن لحظه در دسترس

نباشد فرآیند به حالت انتظار وارد می‌شود. حال اگر منابع محدود مورد درخواست آنها در اختیار فرآیند منتظر دیگری باشد این فرآیند از وضعیت انتظار خارج نمی‌شود، این وضعیت را بن‌بست گویند.

شرایط وقوع بن بست

وضعیت بن‌بست در یک سیستم فقط هنگامی می‌تواند رخ دهد که چهار شرایط کافمن به‌طور هم‌زمان برقرار گردند:

۱- Mutual exclusion (انحصار متقابل): حداقل باید یک منبع به صورت غیر اشتراکی کنترل شود. یعنی فقط یک فرآیند در هر زمان می‌تواند از منبع استفاده کند.
۲- Hold & wait (گرفتن و انتظار): باید فرآیندی موجود باشد که حداقل یک منبع را در اختیار داشته باشد و در حال انتظار برای دریافت منبع دیگر به سر برد که این منبع جدید خود در تصرف فرآیند دیگری است.
۳- Non preemption (منبع انحصاری): یک منبع فقط توسط فرآیندی که آن را در اختیار دارد می‌تواند رها گردد. این رهاسازی بر حسب اراده آن فرآیند و بعد از اتمام کار فرآیند است.
۴- Circular wait (انتظار چرخشی): مجموعه‌ای از فرآیندهای در حال انتظار باید وجود داشته باشند به طوری که P0 در حال انتظار برای منبعی باشد که بوسیله P1 نگه‌داشته می‌شود وP1 در حال انتظار برای منبعی باشد که بوسیله P2 نگه داشته شده است و ... و Pn در حال انتظار برای منبعی باشد که به‌وسیله P0 نگهداری می‌شود.

تمرین:

۱- پیاده‌سازی مسئله خوانندگان و نویسندگان:

فرض کنید یک فرآیند reader و یک فرآیند writer وجود دارند که به ترتیب به خواندن مقدار بافر یا به‌روزرسانی آن می‌پردازند. بین این فرآیندها حافظه مشترکی در نظر بگیرید و در آن مقدار اولیه صفر را بنویسید. توجه داشته باشید که فرآیند writer دسترسی خواندن و نوشتن و فرآیند reader فقط دسترسی خواندن داشته باشد.

- فرآیند writer با هربار دسترسی به بافر مقداری موجود را یک واحد افزایش می‌دهد. writer بعد از دسترسی به بافر پیغامی چاپ می‌کند و در آن شماره فرآیند خودش (PID) و مقدار count را اعلام می‌کند.
- هر reader نیز به طور مداوم مقدار بافر را می‌خواند و در پیغامی شماره فرآیند خودش و مقدار count را اعلام می‌کند.
- شرط پایان این است که مقدار count به یک مقدار بیشینه دلخواه برسد.

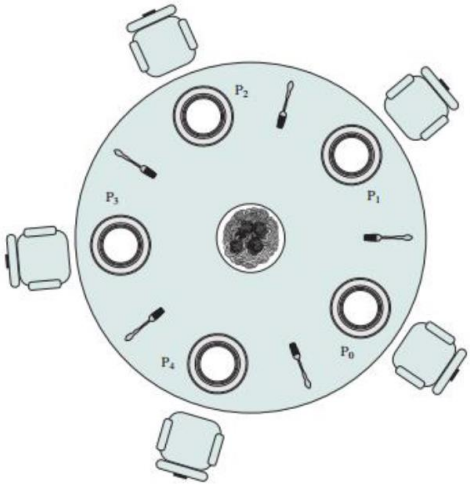
برنامه مربوطه را به‌صورت کامل نوشته و سپس اجرا کنید.

راهنمایی: برای همگام‌سازی فرآیندهای reader و writer می‌توانید از روش‌های همگام‌سازی استفاده کنید. در این صورت وقتی اولین reader به بافر دسترسی می‌یابد، باید آن را lock کند و وقتی آخرین readerکارش تمام شد lock را رها می‌کند. فرآیند writer زمانی می‌تواند مقداری بنویسد که فرآیند reader به بافر دسترسی نداشته باشد و تا اتمام عملیات نوشتن، فرآیند reader قادر به خواندن نیست.

۲- پیاده‌سازی مسئله شام فیلسوفان

این یک مسئله کلاسیک در مبحث همگام‌سازی فرآیندها است. این مسئله یک نمایش ساده از شرایطی است که تعدادی منبع در اختیار تعدادی فرآیند است و قرار است از پیش آمدن بن‌بست یا قحطی جلوگیری شود. میز گردی در نظر بگیرید که ۵ فیلسوف دور آن نشسته‌اند. ظرفی از اسپاگتی در وسط میز قرار گرفته است و ۵ چنگال برای غذا خوردن وجود دارد (بین هر دو نفر یک چنگال قرار دارد.) هر فیلسوف مدتی فکر می‌کند و وقتی گرسنه شد دو چنگال لازم دارد تا بتواند از غذای وسط میز بخورد. طبیعی است که در یک زمان مشخص هر چنگال می‌تواند در اختیار تنها یک فیلسوف باشد و اگر چنگالی در اختیاری فیلسوف کناری باشد امکان در اختیار گرفتن آن وجود نخواهد

داشت. هر فیلسوف پس از اتمام مرحله غذا خوردن خود چنگال را بر روی میز قرار داده و فیلسوف کناری در صورت نیاز می‌تواند از آن استفاده کند.



برنامه مربوطه را به‌صورت کامل نوشته و سپس اجرا کنید. خروجی کد شما می‌تواند به صورت زیر باشد:

```
philosopher 1 is thinking !!
philosopher 1 is eating using chopstick[0] and chopstick[1]!!
philosopher 5 is thinking !!
philosopher 3 is thinking !!
philosopher 3 is eating using chopstick[2] and chopstick[3]!!
philosopher 2 is thinking !!
philosopher 4 is thinking !!
philosopher 1 finished eating !!
philosopher 5 is eating using chopstick[4] and chopstick[0]!!
philosopher 2 is eating using chopstick[1] and chopstick[2]!!
philosopher 3 finished eating !!
philosopher 5 finished eating !!
philosopher 2 finished eating !!
philosopher 4 is eating using chopstick[3] and chopstick[4]!!
```

- ✓ توضیح دهید آیا ممکن است بن‌بست رخ دهد؟ در صورت امکان چگونگی ایجاد آن را توضیح دهید.