دانشکده علوم ریاضی و آمار

مدرس: **دکتر مجتبی رفیعی** نیمسال دوم ۱۴۰۰-۱۴۰۱

اصول سیستمهای عامل

پیاده‌سازی مسئله خواننده_نویسنده

نگارنده: مریم رضائی

۹ تیر ۱۴۰۱

فهرست مطالب

۱ شرح مسئله

مسئله خواننده–نویسنده مسئله حل ناحیه بحرانی تعدادی فرآیند همکار است که داده‌ای را در اشتراک دارند و می‌خواهند (۱) یا بر آن مقداری بنویسند و (۲) یا مقادیر آن را بخوانند، به طوری که در داده ناهمخوانی ایجاد نشود. برای شکل کلی مسئله پارامترهایی را در نظر داریم:

- داده‌ای مشترک میان چند فرآیند موجود است.
- فرآیندها یا نویسنده‌اند و یا خواننده، یعنی فرآیندهای خواننده نمی‌توانند بنویسند.
- زمانی که فرآیند نویسنده‌ای در حال نوشتن است، هیچ فرآیند دیگری نباید به داده دسترسی داشته باشد؛ یعنی فرآیند دیگری بر آن ننویسد، یا مقدار آن را نخواند.
- زمانی که فرآیند خواننده‌ای در حال خواندن است، تنها فرآیندهای خواننده دیگر می‌توانند به داده دسترسی داشته باشند؛ یعنی داده می‌تواند توسط چند فرآیند خوانده شود، اما همزمان نمیتواند فرآیند نویسنده‌ای در آن تغییر ایجاد کند.

در مسئله فوق اولویت اهمیت دارد. اولویت می‌تواند با خوانتده‌ها، با نویسنده‌ها، و یا با ترتیب ورودشان باشد. این امر مسئله را به ۳ نوع زیر تفسیم می‌کند که در الگوریتم و پیاده‌سازی تفاوت‌هایی دارند:

۱. **اولویت با خواننده‌ها:** تا زمانی که خواننده‌ای قصد خواندن دارد به نویسنده‌ها اجازه نوشتن ندهیم.

۲. **اولویت با نویسنده‌ها:** تا زمانی که نویسنده‌ای قصد نوشتن دارد به خواننده‌ها اجازه خواندن داده را ندهیم.

۳. **اولویت ورود:** با صرف نظر از خواننده یا نویسنده بودن فرآیندها به ترتیب ورود به آن‌ها اجازه دسترسی به داده را دهیم.

در ادامه به حل نوع دوم مسئله یعنی اولویت با نویسنده‌ها می‌پردازیم.

۲ حل مسئله

۱.۲ راه کلی

برای حل مسئله خواننده–نویسنده با اولویت برای نویسندگان به موارد زیر احتیاج داریم:

- **شمارنده تعداد نویسنده‌های منتظر:** از آنجا که اولویت را برای نویسندگان قرار داده‌ایم و نمی‌خواهیم تا نویسنده‌ای منتظر است به خوانندگان اجازه دسترسی به داده را دهیم، با شمارنده‌ای تعداد نویسندگان که متوقف شده‌اند را ذخیره می‌کنیم و هر بار که کار فرآیندی از ناحیه بحرانی گذشت، با بررسی تعداد این شمارنده تعیین می‌کنیم که آیا میخواهیم نویسندگان را بیدار کنیم یا خوانندگان را.

- **سمافورهای اجازه به خواننده یا نویسنده:** برای بررسی دو شرط اجازه دسترسی نویسندگان و یا خوانندگان به داده و بیدار کردن فرآیند نوع مورد نظر پس از پایان ناحیه بحرانی هر فرآیند، نیاز به دو سمافور با صف‌های مجزا داریم که به صورت دو شرط عمل می‌کنند. شمارنده تعداد نویسندگان منتظر (مورد قبل) در اصل شمارنده سمافور نویسندگان آن که فرآیندهای در صف خود را در نظر دارد.

- **سمافور mutex**: برای حفظ مقادیر و شمارنده‌های مشترک، قفلی نیاز داریم تا دسترسی به تغییر در شمارنده را محدود کرده و در هر لحظه یک فرآیند بر آن‌ها عمل ایجاد کند.

۲.۲ مقدمات و فروض پیاده‌سازی

پیاده‌سازی این راه حل را با زبان پایتون انجام می‌دهیم. برای این کار از ۳ ماژول اصلی پایتون که با زبان پایتون خودکار نصب می‌شوند استفاده می‌کنیم:

۱. **ماژول threading**: این ماژول برای ایجاد نخ‌هایی از برنامه موجود است که اجازه استفاده از سمافورهای از پیش تعیین شده را می‌دهد. آنگاه نخ‌ها در این ماژول به صورت قبضه‌ای و به ترتیب ورود (یعنی با الگوریتم RR) اجرا می‌شوند. همچنین ماژول برای یک قفل تعریف شده، اجازه تعریف شروطی را می‌دهد که به مانند قفل‌های درونی عمل می‌کنند.

۲. **ماژول logging**: به قصد مشاهده شیوه پیش‌روی برنامه با استفاده از این کتابخانه، پیام‌هایی حاوی عمل انجام شده در بخش های اصلی فرآیندها (زمان شروع، ورود و خروج از ناحیه بحرانی، و عمل درون ناحیه بحرانی) قرار می‌دهیم.

۳. **ماژول random**: با کمک این کتابخانه تعیین نوع رشته تشکیل شده برنامه (رشته نویسنده یا خواننده باشد) و مقادیر نوشته شده توسط فرآیندهای نویسنده را تصادفی می‌کنیم.

همچنین در نظر داریم که از یک قفل سمافور تعداد تعیین شده ای می‌توانند گذر کنند و به قصد داشتن قفلی به دور ناحیه بحرانی به طوری که برای نویسندگان تنها یک نخ عبور کند اما برای خوانندگان، در صورت امکان، همگی به هر تعدادی عبور

کنند و به داده دسترسی داشته باشند، نیاز به تغییر در تعریف قفل‌ها داریم. بنابراین می‌توانیم با استفاده از قفل‌های از پیش تعیین شده، کلاس قفل جدیدی تعریف کنیم که برای ورود به ناحیه بحرانی دو نوع تابع acquire داشته باشد که هر کدام برای نویسندگان و خوانندگان باشد و فرآیندها را قبل از ورود به ناحیه بحرانی متوقف و بررسی کند و پس از تایید، تعداد مورد نظر را رها کند تا وارد ناحیه بحرانی شوند.

۳.۲ توضیح برنامه حاصل

برنامه حاصل که در فایل reader-writer.py قابل دسترسی‌ست، دارای ۵ بخش زیر می‌باشد:

۱. **بدنه اصلی:** با استفاده از تابع انتخاب تصادفی، ۱۰ نخ تشکیل می‌دهیم که یا از نوع خواننده و یا نویسنده‌اند. در تشکیل هر نخ، نام آن را با توجه به نوع آن و چندمین بودن از آن نوع تعیین می‌کنیم. هر نخ را پس از تشکیل اجرا کرده و در آخر پس از پایان تمامی نخ‌ها آن‌ها را join می‌کنیم.

۲. **بافر مشترک:** برای ذخیره داده اشتراکی و تعداد نخ‌هایی که در هر لحظه به آن دسترسی دارند، کلاسی تعریف می‌کنیم که در آن با بررسی نوع نخ‌های در حال استفاده از آن تعیین می‌کند آیا میتوان به نویسنده یا خواننده‌ای که درخواست دسترسی می‌دهد، اجازه داد یا خیر.

۳. **قفل خواننده‌ـنویسنده:** با استفاده از قفل‌های پایتون، کلاسی قفلی جدید تعریف می‌کنیم که ۳ تابه اصلی دارد: (۱) تابع درخواست نویسنده، (۲) تابه درخواست خواننده، (۳) تابع رهایی هر دو. این توابع توسط قفلی به ناحیه بحرانی تبدیل شده و در هر کدام با بررسی دو شرط تعریف شده و سوال از بافر، نخ‌ها را متوقف کرده و یا رها می‌کنیم.

۴. **تابع نویسنده:** در این تابع که در نخ نوع نویسنده فراخوانی می‌شود، قبل از ورود به ناحیه بحرانی، قفل تعریف شده را گرفته و در صورت اجازه گذر، عددی تصادفی از میان ۰ تا ۱۰۰ را به آرایه داده‌های اشتراکی اضافه می‌کنیم.

۵. **تابع خواننده:** به مانند تابع نویسنده از قفل تعریف شده استفاده می‌کنیم تا وارد ناحیه بحرانی شده و سپس از آن خارج شویم. خواننده در ناحیه بحرانی شروع به خواندن تک تک اعضای آرایه بافر می‌کند، که به علت قبضه‌ای بودن اجرا و اجازه دسترسی چند خواننده به داده، خواندن عناصر آرایه یکی‌ـچند در میان بین خواننده‌های در حال اجرا اتفاق می‌افتد.