پروژه‌ی نهایی سیستم عامل - پاییز ۱۴۰۲

یکی از پروژه‌های پرکاربرد در صنعت نرم‌افزار، سیستم‌های صف هستند. کار صف در واقع حل کردن مسئله‌ی تولیدکننده-مصرف کننده در یک محیط توزیع‌شده است. در یک صف تعداد زیادی تولیدکننده‌ی هم‌روند داریم که اقدام به تولید مسیج جدید می‌کنند. همچنین تعداد زیادی مصرف‌کننده‌ی همروند داریم که (در صورت وجود) پیام‌های داخل صف را می‌خوانند. در زمانی که روند تولید پیام‌ها از مصرف آن‌ها بیشتر باشد طول صف بیشتر می‌شود و زمانی که روند مصرف پیام‌ها بیشتر باشد طول صف کم می‌شود تا به صفر برسد. در زمانی که طول صف صفر باشد مصرف‌کننده‌ها باید صبر کنند تا پیام جدید تولید شود.

در این پروژه از شما خواسته می‌شود که یک صف پیاده‌سازی کنید که امکان اضافه کردن پیام به ابتدا و خواندن پیام از انتها را داشته باشد. همچنین امکان گرفتن گزارش از تعداد پیام‌های داخل صف نیز باید وجود داشته باشد. نکته‌ی مهم این پیاده‌سازی هندل کردن هم‌روندی به شکل مناسب است.

امکانات مورد نیاز

- تابع send_msg که یک رشته به عنوان ورودی گرفته و آن را در صف قرار دهد.
- تابع get_msg که در صورت وجود یک پیام در صف آن را برمی‌گرداند و در غیر این صورت این قدر منتظر می‌ماند تا یک پیام در صف قرار بگیرد.
- تابع get_msg_nb که مانند مورد قبل است با این تفاوت که در صورتی که پیامی در صف موجود نباشد به سرعت باز می‌گردد و معطل نمی‌شود (یعنی non blocking است.)
- تابع stats که موارد زیر را در یک کلاس/استراکت باز می‌گرداند: تعداد پیام‌های داخل صف، مجموع طول پیام‌های داخل صف، مقدار مموری‌ای که برنامه‌ی شما مصرف می‌کند (برای این منظور می‌توانید از سیستم کال‌های سیستم‌عامل خود استفاده کنید.)
- سازنده‌ی صف، باید این امکان را داشته باشد که **صفی با طول نامحدود و یا صفی با طول محدود** بسازد. زمانی که صف با طول محدود پر شود، اضافه کردن پیام جدید ناموفق خواهد بود.

نیازمندی‌های هم‌روندی

- برنامه‌ی شما باید امکان هندل کردن چند تولیدکننده و مصرف‌کننده به طور همزمان را داشته باشد.
- در صورتی که فقط یک پیام در صف است و همزمان دو مصرف‌کننده درخواست می‌کنند، برنامه‌ی شما باید به شکل مناسب مسئله را حل کند و پیام را فقط در اختیار یکی از آن‌ها قرار دهد.
- در صورتی که چند مصرف‌کننده منتظر پیام هستند و یک پیام جدید وارد صف می‌شود باید به درستی هندل شود که فقط یکی از مصرف‌کننده‌ها پیام را مصرف کند.
- در صورتی که چندین مصرف‌کننده داریم که در زمان‌های مختلفی منتظر شده‌اند، اولویتی بین آن‌ها قائل نیستیم.
- در صورتی که صف دارای چندین پیام باشد و همزمان چند تولیدکننده و چند مصرف‌کننده بخواهند اقدام کنند، نباید تولیدکننده‌ها مانع مصرف‌کننده‌ها و مصرف‌کننده‌ها مانع تولیدکننده‌ها شوند. در **حالت ساده‌تر، اگر فقط یک تولید کننده و یک مصرف کننده داشته باشید باید بتوانند همزمان به کار خود ادامه دهند و لاک نشوند.**
- آیتم‌هایی که زودتر وارد صف شده‌اند باید الزاما زودتر از صف خارج شوند.
- در صورتی که همزمان دو تولیدکننده اقدام به تولید کنند، ترتیب پیام‌هایشان در صف اهمیتی ندارد.

محدودیت‌ها

- برای پیاده‌سازی از یکی از زبان‌های برنامه‌نویسی سی، سی‌پلاس‌پلاس، پایتون و یا جاوا استفاده کنید.
- برای پیاده‌سازی این پروژه از مکانیسم‌های ابتدایی هم‌روندی یعنی تردها و پروسس‌ها استفاده کنید و مواردی مانند thread safe data structureها، threadpool و غیره مجاز نیست.
- برای کنترل هم‌روندی نیز تنها مجاز به استفاده از امکانات ساده‌ی متغیرهای اتمیک و mutex و semaphore هستید. در صورتی که این دو در زبان برنامه‌نویسی مورد نظر شما امکانات بیشتر از آن چه در این درس آمده نیز دارند (مثلا non blocking unlock یا fairness یا check is locked یا read write lock) مجاز به استفاده از آن‌ها نیستید.

موارد امتیازی

- وجود تاپیک‌های مختلف برای ارسال پیام
- صف دارای اولویت
- پیاده‌سازی time to live یک پیام در سیستم
- محدود کردن حجم آیتم‌های داخل صف علاوه بر تعداد
- وجود سیاست‌های مختلف برای هندل کردن پر شدن صف، مثلا حذف قدیمی ترین پیام و یا جدیدترین پیام و یا یک پیام تصادفی
- امکان ارسال ack توسط مصرف‌کننده در زمانی که پردازشش روی داده تمام شد. در صورت عدم دریافت ack تا مدتی باید پیام مجددا به صف برگردد (به عنوان قدیمی‌ترین مورد)
- امکان ذخیره شدن پیام‌ها و صف‌ها در دیسک و بازگردانی در اجرای بعدی

آن‌چه باید تحویل دهید

این پروژه تحویل خواهد داشت و در زمان تحویل باید موارد زیر را ارائه دهید:

- برنامه‌ی خودتان که امکانات خواسته‌شده دارد.
- **تست‌هایی که برنامه‌ی خودتان را استفاده می‌کند و از قابلیت‌های آن استفاده می‌کند.**
- گزارش مختصری از روش‌های پیاده‌سازی خود برای کنترل هم‌روندی، مثلا تعداد لاک‌هایی که به ازای هر عملیات گرفته می‌شوند و تعداد و لیست کل لاک‌ها و سمافورهای برنامه.