دانشگاه علم و صنعت

دانشکده مهندسی کامپیوتر

درجه تحصیلی: کارشناسی

# گزارشکار تمرین5 OS

گردآورنده:

پرنیان شاکریان - 99400064

استاد:

دکتر انتظاری

سال تحصیلی: خرداد ۱۴۰۲

خلاصه:

در تمرین پنجم سیستم عامل قصد داریم مبحث Inter-process communication (IPC) را بررسی کنیم. اگر بخواهیم به طور خلاصه توضیح دهیم، (IPC) به تکنیک‌هایی اشاره دارد که توسط process برای تبادل اطلاعات، هماهنگ کردن فعالیت‌ها و همگام‌سازی عملیات آنها استفاده می‌شود. (IPC) اجازه می دهد تا processهای در حال اجرا بر روی یک سیستم با یکدیگر ارتباط برقرار کرده و مشارکت کنند.

روش های (IPC):

۱. Pipes: شکل ساده‌ای از IPC را ارائه می دهند که در آن داده از طریق یک کانال ارتباطی یک طرفه بین فرآیندها منتقل می شود. یک فرآیند، داده ها را در Pipe نوشته و فرآیند دیگر آن را می خواند.
۲. Shared Memory: به چندین فرآیند اجازه میدهد تا به بخش حافظه‌ای که در آنها مشترک است دسترسی داشته باشند. فرآیندها می توانند با mapping کردن ناحیه‌ای از حافظه مشترک در فضای آدرس خود، آن را خوانده و بنویسند و اشتراک گذاری داده ها را فراهم کنند. (مکانیسم‌های سمافور یا mutex اغلب برای هماهنگ کردن دسترسی به حافظه مشترک استفاده می‌شوند.)
۳. signals: راهی برای processها به منظور ارسال notification یا قطع سیگنال به processهای دیگر است. از سیگنال‌ها می‌توان برای مدیریت انتقال اطلاعات بین processها استفاده کرد، اما از نظر داده‌هایی که می‌توانند حمل کنند محدود هستند.

شرح کلی: برای حل سوالات تمرین در ابتدا لازم است کلیه کتابخانه‌های مورد نیاز را وارد کنیم. کلیه کتابخانه های مورد استفاده در این تمرین نمایش داده شده است.

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <pthread.h>
#include <semaphore.h>
#include <unistd.h>
#include <fcntl.h>
#include <sys/types.h>
```

1. Implement a multi-threaded program in C that simulates a producer-consumer scenario for IPC.

ابتدا کتابخانه‌های مورد نیاز خود را وارد خواهیم کرد. (در ابتدا داک اشاره شده است). در ادامه ثابت‌های BUFFER_SIZE و MAX_ITEMS را با مقادیر مورد نظر خود تعریف میکنیم. BUFFER_SIZE نشان دهنده اندازه shared buffer و MAX_ITEMS تعداد اقلامی که هر producer تولید میکند را نشان میدهد. در ادامه ساختار shared_buffer را تولید میکنیم که نشان دهنده بافر مشترک بین producers و consumers است و فیلدهای زیر را شامل میباشد:

۱. buffer: یک آرایه عدد صحیح به اندازه BUFFER_SIZE برای نگهداری اقلام تولید شده.
۲. in و out: برای پیگیری موقعیت‌ها در بافر.
۳. mutex، emty و full: اشاره‌گر به سه سمافور مورد استفاده برای همگام سازی.

ساختار thread_args آرگومان هایی را که باید به رشته های producer و consumer منتقل شوند را نشان میدهد که شامل دو فیلد است:

۱. id: شناسه رشته.
۲. sb: اشاره گر به بافر مشترک.

کد ما از ۳ تابع اصلی تشکیل شده است. تابع producer رفتار یک رشته تولید کننده را نشان میدهد به این صورت که یک آرگومان اشاره گر میگیرد که به thread_args فرستاده میشود و pointer بافر مشترک و ID رشته را از ساختار arg بازیابی میکند. سپس رشته producer برای تولید MAX_ITEMS وارد حلقه میشود که یک مقدار تصادفی را با استفاده از rand% 1000 ایجاد میکند. سپس به ترتیب مراحل زیر را برای افزودن آیتم به بافر مشترک انجام میدهد:

۱. منتظر سمافور خالی می‌ماند تا مطمئن شود فضای موجود در بافر برای آیتم جدید وجود دارد.
۲. منتظر مانده تا سمافور mutex به mutual exclusion دست یابد و از critical section محافظت کند.
۳. ایتم را در موقعیتی که با sb->in نشان داده به بافر اضافه کرده و با افزایش آن و گرفتن مدول BUFFER_SIZE، شاخص in را آپدیت میکند.
۴. پیامی را چاپ میکند که نشان می دهد کدام producer کالا و ارزش آن را تولید کرده است.
۵. سمافور mutex را آزاد کرده تا به سایر رشته ها اجازه دسترسی به critical sectionرا بدهد.
۶. به سمافور full سیگنال میدهد که یک مورد تولید شده و برای مصرف در دسترس است.

۷. در آخر با استفاده از usleep(rand %1000) برای شبیه سازی زمان های مختلف تولید، تاخیر معرفی میکند.

تابع consumer رفتار یک رشته مصرف کننده را نشان میدهد که یک آرگومان اشاره گر میگیرد که به thread_args فرستاده می شود و بافر مشترک و شناسه رشته را از ساختار arg بازیابی میکند. رشته consumer برای مصرف اقلام از بافر مشترک وارد یک حلقه بی نهایت می شود و برای بازیابی یک آیتم از بافر به ترتیب مراحل زیر را انجام میدهد:

۱. منتظر می‌ماند تا سمافور اطمینان پیدا کند که حداقل یک مورد برای مصرف در بافر وجود دارد.
۲. منتظر می‌ماند تا سمافور mutex به طرد متقابل دست یافته و از بخش بحرانی محافظت کند.
۳. آیتم را از بافر در موقعیتی که با sb->out نشان داده شده است بازیابی کرده و با افزایش آن و گرفتن مدول BUFFER_SIZE، شاخص خروجی را به روز می‌کند.
۴. پیامی را چاپ کرده که نشان می دهد کدام consumer کالا را مصرف کرده و ارزش آن را نشان می دهد.
۵. سمافور mutex را آزاد می کند تا به سایر رشته ها اجازه دسترسی به بخش بحرانی را بدهد.
۶. به سمافور خالی سیگنال می دهد که اکنون یک ناحیه خالی در بافر برای تولید وجود دارد.
۷. در آخر با استفاده از usleep(rand %1000) برای شبیه سازی زمان های مختلف تولید، تاخیر معرفی میکند.

تابع main:

- shm_fd: یک توصیفگر فایل است که برای حافظه مشترک استفاده می شود.
- sb: اشاره گر به بافر مشترک است.

تابع با ایجاد یک shared memory object با استفاده از shm_open و پرچم‌های O_CREAT | O_RDWR شروع میشود. این یک منطقه حافظه مشترک با مجوزهای خواندن و نوشتن ایجاد میکند. سپس اندازه shared memory object را با استفاده از ftruncate به اندازه shared_buffer تنظیم می کند. این تضمین می کند که حافظه کافی برای نگهداری بافر مشترک تخصیص داده شده است. در مرحله بعد، shared memory object را با استفاده از mmap به نشانگر sb نگاشت کرده و به بافر مشترک اجازه میدهد تا توسط چندین فرآیند قابل دسترسی شود. شاخص های ورودی و خروجی بافر مشترک به ۰ مقداردهی اولیه می شوند. این شاخص ها موقعیت‌هایی را در بافر که در آن تولیدکنندگان اقلام را تولید و مصرف کنندگان اقلام را مصرف می کنند، پیگیری

می کند. سپس کد سه سمافور (mutex، empty و full) را با استفاده از sem_open ایجاد میکند. اولین سمافور، mutex، با مقدار اولیه ۱ مقداردهی اولیه می‌شود تا برای بخش‌های بحرانی حذف متقابل فراهم شود. سمافور empty با مقدار اولیه BUFFER_SIZE مقداردهی می شود که تعداد اسلات های خالی موجود در بافر را نشان می دهد. سمافور full با مقدار اولیه ۰ مقداردهی می شود که تعداد اقلام موجود برای مصرف را نشان می دهد. پس از راه‌اندازی بافر و سمافورهای مشترک، کد با استفاده از pthread_create و pthread_join، رشته‌های producer وconsumer را ایجاد و اجرا می‌کند.

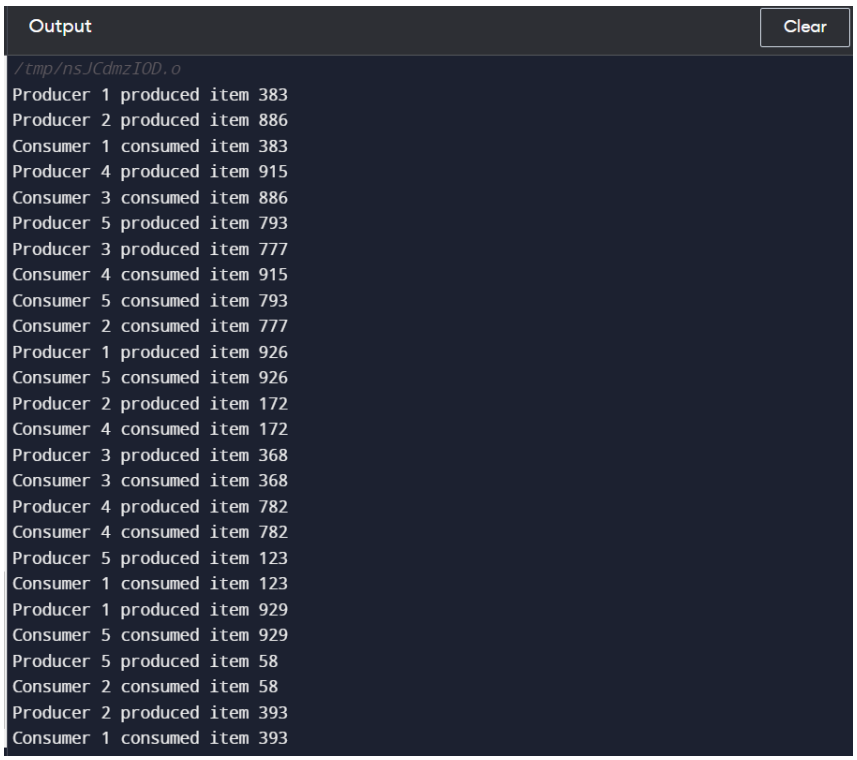
- pthread_t producers, consumers: آرایه هایی را برای ذخیره ID رشته‌های consumer و producer اعلام می کند.
- thread_args producer_args, consumer_args: آرایه‌هایی را برای ذخیره آرگومان‌هایی که باید به رشته‌های producer و consumer ارسال شوند، اعلام می‌کند.

حلقه for رشته های producer و consumer را ایجاد می کند. در هر تکرار حلقه:

- id ساختارهای producer_args و customers_args با مقدار i + 1 تنظیم شده است که نشان دهنده شناسه رشته است.
- sb ساختارهای producer_args و customers_args روی pointer بافر مشترک sb تنظیم شده است.
- pthread_create برای ایجاد رشته های producer و consumer فراخوانی می شود.

پس از ایجاد thread ها، یک حلقه for دیگر با pthread_join استفاده می شود تا منتظر خاتمه باشد. این تضمین میکند که thread اصلی منتظر می ماند تا تمام thread های دیگر قبل از ادامه، اجرای خود را کامل کنند. در نهایت، حافظه مشترک و سمافورهای نامگذاری شده پاک می شوند و تابع main EXIT_SUCCESS را برای نشان دادن اجرای موفقیت آمیز برمیگرداند. بخشی از خروجی را در داک اورده‌ایم (در صورت عدم مشاهده خروجی لطفا چند بار آن را run کنید با هر بار ران کردن اعداد رندوم تولید میشوند)

خروجی اول




خروجی دوم