

به نام ایزد یکتا



دانشگاه صنعتی امیرکبیر
(پلی تکنیک تهران)



دانشکده مهندسی کامپیوتر

آزمایش ششم درس آزمایشگاه سیستم عامل

استاد: مهندس قاسمی

تهیه کننده: بردیا اردکانیان

۹۸۳۱۰۷۲

آزمایش اول)

برای این بخش، ابتدا برای ایجاد پردازش‌های reader و writer از fork می‌خواهیم استفاده کنیم. که این پردازش‌ها همگی به یک memory shared دسترسی دارند پس یک shared memory ایجاد می‌کنیم و مشخص می‌کنیم که روی آن می‌خواهیم از mutex استفاده کنیم. حال mutex چیست؟ mutex مانند یک قفل است که برای تردها استفاده می‌شود، وقتی یک ترد آن را قفل می‌کند و وارد section critical می‌شود، ترد دیگری نمی‌تواند وارد آن قسمت شود و آن دیتا را تغییر دهد تا زمانی که قفل توسط همان ترد قبلی باز شود.

در اینجا چون از fork استفاده کردیم و می‌خواهیم از مکانیزم mutex نیز استفاده کنیم، shared memory را به صورت آرایه ای از استراکت‌ها تعریف می‌کنیم که در آن مقدار counter، تعداد پردازش‌های reader و mutex را نگه داری می‌کنیم.

در واقع reader همان mutex up و writer به عبارتی mutex down است.

حال بعد از ایجاد readers ها و writer، به این صورت عمل می‌کنیم که هنگامی که writer بخواهد بنویسد mutex را قفل می‌کند و پس از انجام کار آن را آزاد می‌کند. و در readers، اولین خواننده mutex را قفل می‌کند و آخری آزاد می‌کند و به همین دلیل است که به تعداد readers نیاز داریم و آن را داخل استراکت تعریف می‌کنیم.

آزمایش دوم)

در این بخش، مثل بخش قبلی از `mutex` استفاده می‌کنیم به این صورت که ۵ ترد می‌سازیم که هر کدام نماینده یک فیلسوف است. حال برای هر چاپ استیک می‌خواهیم یک قفل قرار دهیم. برای اینکار از `t_mutex_pthread` استفاده می‌کنیم و یک آرایه ۵ تایی از آن می‌سازیم. در هر ترد کاری که می‌کنیم این است که آن فیلسوف، چاپ استیک‌های دو طرف خودش را بر میدارد و آن‌ها را قفل می‌کند تا بقیه از آن‌ها نتوانند استفاده کنند. سپس غذا می‌خورد و پس از خوردن قفل را آزاد می‌کند. درواقع `t_mutex_pthread` کاری که می‌کند این است که آن ترد با قفل کردن اجازه نمی‌دهد تردهای دیگر به آن دسترسی پیدا کنند.

همچنین می‌توانیم از مکانیزم `semaphore` برای قفل کردن استفاده کنیم.