پاسخ‌نامه تمرین ششم درس سیستم‌های عامل

دکتر زرندی

پاییز ۱۴۰۰

1- به سوالات زیر پاسخ کوتاه دهید:

الف) در ساختار کلاس *MulticastSocket* جاوا از چه نوع سوکت دیگری استفاده شده است؟

ب) در صورتی که یک فرایند از پورت *1600* میزبان (*host*) اتصالی با یک وب سرور برقرار کرده باشد و همزمان فرایند دیگری روی همان میزبان قصد برقراری ارتباط با همان وب سرور را داشته باشد، چه محدودیتی در اختصاص پورت به فرایند جدید داریم؟

ج) چرا استفاده از سوکت، روشی سطح پایین برای ارتباط بین فرایندهای توزیع شده در نظر گرفته میشود؟ چه روش سطح بالاتری را برای این منظور می‌توان نام برد؟

الف) جاوا سه نوع مختلف سوکت در اختیار ما می‌گذارد: سوکت‌های اتصال‌گرا (*TCP*) که با استفاده از کلاس *Socket* پیاده‌سازی شده‌اند، سوکت‌های غیراتصالی (*UDP*) که از کلاس *DatagramSocket* استفاده می‌کنند و کلاس *MulticastSocket* که زیرمجموعه کلاس *DatagramSocket* بوده و به داده‌ها اجازه می‌دهد که به چندین مقصد فرستاده شوند.

ب) هر اتصال باید یک پورت منحصر به فرد داشته باشد. بنابراین اگر یک فرآیند دیگر روی همان میزبان قصد برقراری ارتباط با همان وب سرور را داشته باشد، یک شماره پورت بزرگ‌تر از ۱۰۲۴ و نامساوی ۱۶۰۰ به آن اختصاص داده خواهد شد. به این ترتیب همه‌ی اتصالات دارای جفت سوکت‌های متمایز خواهند بود.

ج) ارتباطاتی که از سوکت استفاده می‌کنند با اینکه مرسوم و کارا هستند، یک شکل سطح پایین از ارتباط بین فرآیندهای توزیع‌شده محسوب می‌شوند. یک دلیل این است که سوکت‌ها فقط امکان تبادل یک جریان بدون ساختار از داده بین دو ریسمان را فراهم می‌کنند. این وظیفه برنامه سرور یا کلاینت است که یک ساختار بر داده‌ها تحمیل کند. برای ارتباطات سطح بالاتر می‌توان از فراخوانی رویه راه دور (*RPC*) استفاده کرد.

۲- در هر یک از شرایط زیر توضیح دهید استفاده از کدام نوع از *pipe*ها مناسب است؟(*ordinary pipe, named pipe*)

الف) پس از پایان ارتباط دیگر نمی‌خواهیم به *pipe* دسترسی داشته باشیم.

ب) می‌خواهیم از *pipe* بین شبکه‌ای از فرایندها استفاده کنیم.

ج) می‌خواهیم امکان ارتباط دو طرفه (*bidirectional*) فراهم باشد.

د) می‌خواهیم تنها والد و فرزند به آن دسترسی داشته باشند.

لوله‌های عادی (*ordinary pipes*) یک طرفه هستند و به یک رابطه والد-فرزندی بین دو فرآیند نیاز دارند. در هر دو سیستم یونیکس (*UNIX*) ویندوز (*Windows*)، هنگامی که فرآیندها به ارتباط خود خاتمه می‌دهند و پایان می‌یابند، لوله عادی نیز از بین می‌رود. لوله‌های نامدار (*named pipes*) یک ابزار ارتباطی قدرتمندتر هستنند. با استفاده از آن‌ها ارتباط می‌تواند دو طرفه باشد و به رابطه والد-فرزندی نیز نیازی نیست. هنگامی که یک لوله نامدار ساخته می‌شود، چندین فرآیند می‌توانند از آن برای برقراری ارتباط استفاده کنند.

الف) لوله عادی

ب) لوله نامدار

ج) لوله نامدار

د) لوله عادی

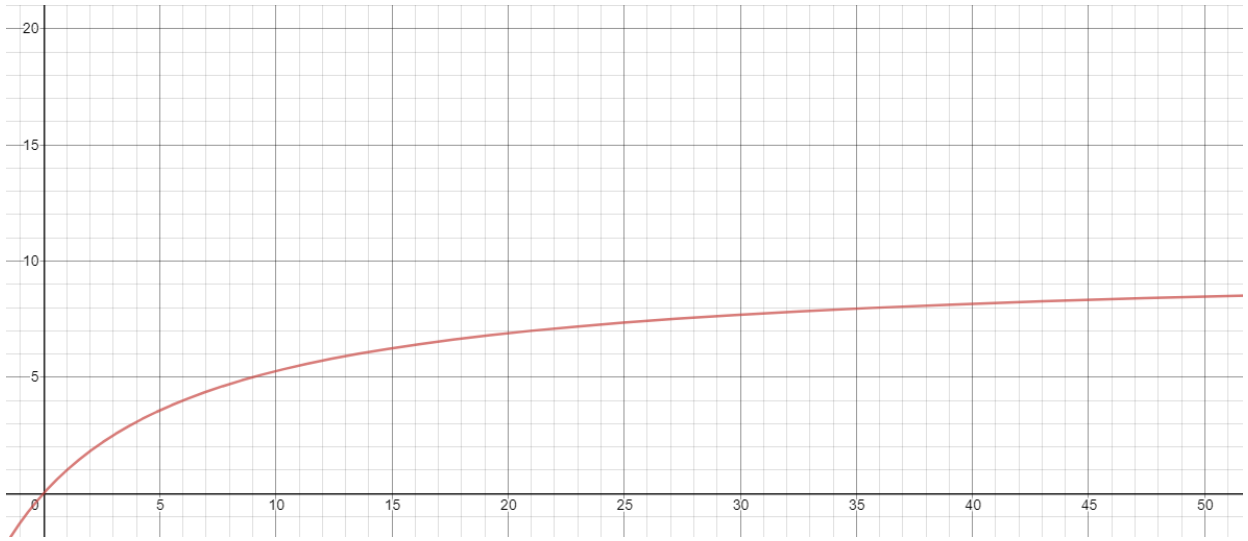
۳- یک الگوریتم داریم که ۹۰٪ قابلیت موازی سازی دارد. برای اینکه به ۸۰٪ از حداکثر میزان تسریع تئوری دست پیدا کنیم، باید حداقل از چه تعداد ریسمان در اجرای این الگوریتم استفاده کرد؟

Amdahl's Law:

$$S_{\text{latency}}(s) = \frac{1}{(1-p) + \frac{p}{s}}$$

Maximum Theoretical Speedup:

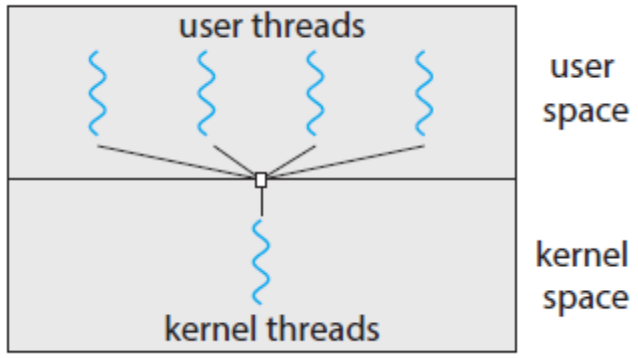
$$\lim_{s \to \infty} S_{\text{latency}}(s) = \frac{1}{1-p}$$



$$\frac{1}{(1-0.9) + \frac{0.9}{s}} = 0.8 \times \frac{1}{1-0.9}$$

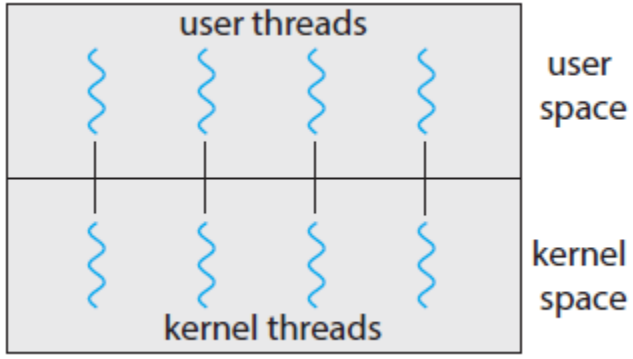
$$s = 36$$

۴- مدل‌های مختلفی برای ارتباط میان ریسمان‌های هسته و ریسمان‌های کاربر وجود دارد:
الف) در چه حالتی مدیریت ریسمان‌ها بهینه‌تر می‌باشد؟ در چه روشی محدودیتی برای تعداد ریسمان‌های هسته وجود ندارد؟ آیا این روش‌ها در سیستم‌عامل‌های امروزی کاربرد دارند؟ چرا؟



در مدل چند به یک مدیریت ریسمان‌ها بهینه می‌باشد.
زیرا توسط کتابخانه ریسمان موجود در فضای کاربر انجام می‌شود.

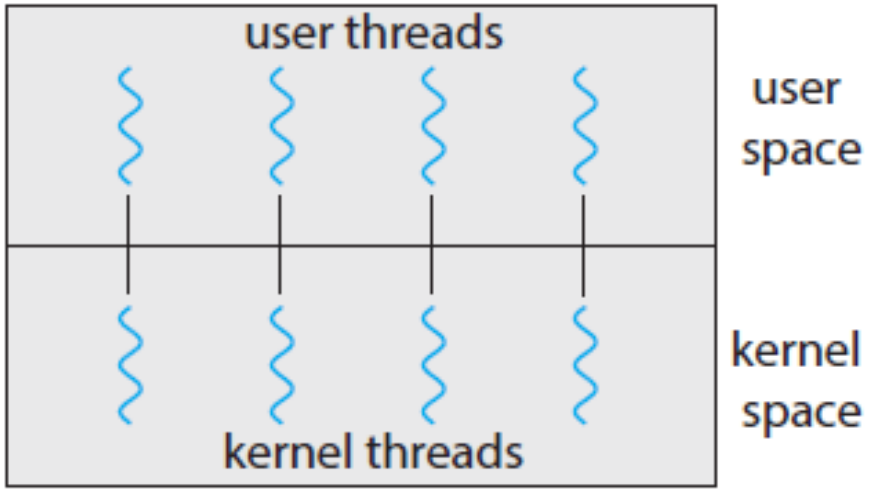
در واقعیت تعداد ریسمان‌های هسته نمیتواند از حدی بیشتر شود اما در مدل یک به یک طبق تعریف به ازای هر ریسمان کاربر یک ریسمان هسته ایجاد می‌شود. اگر کاربر تعداد زیادی ریسمان ایجاد کند، می‌تواند باعث کند شدن سیستم شود.



امروزه مدل چند به یک متداول نیست زیرا زیرا قادر به بهره‌وری از پردازنده‌های چندهسته‌ای نیست. در عوض از مدل یک به یک در بیشتر سیستم‌عامل‌های امروزی استفاده می‌شود، زیرا اجازه اجرای تعداد زیادی ریسمان را می‌دهد. همچنین با افزایش تعداد هسته‌های پردازشی دیگر محدود کردن تعداد ریسمان‌های هسته اهمیت چندانی ندارد.

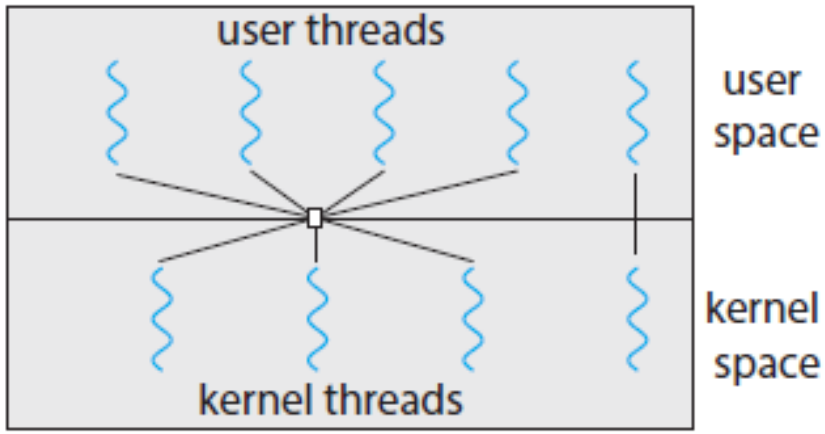
ب) در مدل یک به یک چه احتیاطی باید رعایت شود؟ چرا؟

در مدل یک به یک، کاربر در مورد تعداد ریسمان‌هایی که ایجاد می‌کند احتیاط کند.
زیرا به ازای هر ریسمان کاربر یک ریسمان هسته ایجاد می‌شود.
تعداد زیاد ریسمان‌های هسته می‌تواند موجب کند شدن سیستم شود.



ج) مزیت‌های مدل چند به چند چیست؟ چرا در سیستم‌های عامل امروزی از این روش استفاده نمی‌شود؟

در مدل چند به چند، بر خلاف مدل چند به یک، می‌توان از چندین ریسمان هسته استفاده کرد که باعث هم‌روندی بیشتر و بهره‌وری از هسته‌های پردازشی می‌شود.



همچنین بر خلاف مدل یک به یک، با افزایش تعداد ریسمان‌های کاربر سیستم دچار مشکل نشده و نیازی نیست که کاربر نگران این موضوع باشد.

پیاده‌سازی مدل چند به چند از بقیه مدل‌ها پیچیده‌تر می‌باشد. علاوه بر آن با افزایش تعداد هسته‌های پردازشی در سیستم‌های امروزی، دیگر نیازی به محدود کردن تعداد ریسمان‌های هسته نیست. به همین دلیل اکثر سیستم‌عامل‌های امروزی از مدل یک به یک استفاده می‌کنند.