سیستم‌های عامل

نیم‌سال اول ۰۳-۰۲

دانشکده‌ی مهندسی کامپیوتر

شماره دانشجویی: ۹۹۱۰۵۵۶۱

شایان صالحی

تمرین دوم

پاسخ مسئله‌ی ۱.

آ

نادرست، Context switch ممکن است به دلایل مختلفی از جمله تغییر در اولویت‌های برنامه‌ریزی وقتی که یک پردازه (process) در انتظار منابع است، یا برنامه‌ریزی زمانی برای اطمینان از توزیع منصفانه زمان پردازنده بین پردازه‌ها، رخ دهد. این فرآیند همیشه به صورت ناخواسته و در زمان‌های نامشخص رخ نمی‌دهد.

ب

نادرست، در حالت عمومی، هر ریسه (thread) دارای استک (stack) اختصاصی خود است و به طور مستقیم نمی‌تواند به استک سایر ریسه‌ها دسترسی داشته باشد. دسترسی به داده‌های استک یک ریسه توسط ریسه‌ای دیگر نیازمند مکانیزم‌های خاصی مانند اشتراک‌گذاری حافظه است.

ج

نادرست، تابع execv در لینوکس برای جایگزینی فعلی پردازه با یک پردازه جدید استفاده می‌شود، نه ایجاد یک پردازه جدید. این تابع، کد موجود در یک پردازه را با یک برنامه جدید جایگزین می‌کند. برای ایجاد یک پردازه جدید، معمولاً از تابع fork استفاده می‌شود، که یک کپی از پردازه فعلی را ایجاد می‌کند.

د

درست، در سیستم‌های عاملی مانند UNIX و لینوکس، یک پردازه می‌تواند چندین file descriptor داشته باشد که به یک file description واحد اشاره می‌کنند. این امر به ویژه در مواردی مانند استفاده از توابعی چون fork یا دوباره باز کردن یک فایل مشاهده می‌شود.

ه

نادرست، اگر تردها فقط برای خواندن داده‌ها از آرایه استفاده کنند. در مواردی که چندین ریسه (thread) فقط برای خواندن داده‌ها از یک منبع مشترک استفاده می‌کنند و هیچ نوشتنی در آن صورت نمی‌گیرد، race condition به وجود نمی‌آید و نیازی به استفاده از مکانیزم‌های سینکرونایزیشن نیست. Race condition زمانی رخ می‌دهد که چندین ریسه به طور همزمان داده‌ها را می‌خوانند و می‌نویسند.

و

درست، در سیستم‌های عامل مدرن، فضای آدرس هر پردازه از دیگر پردازه‌ها مجزا و محافظت شده است. یک پردازه نمی‌تواند مستقیماً فضای حافظه یا آدرس‌دهی یک پردازه دیگر را تغییر دهد یا به آن دسترسی پیدا کند.

ز

نادرست، زمان انتظار در الگوریتم Round Robin نسبت به FCFS بستگی به عوامل متعددی دارد، از جمله طول دوره زمانی (time slice) و خصوصیات ویژه بار کاری. در برخی موارد، زمان انتظار در Round Robin می‌تواند کمتر از FCFS باشد، به خصوص اگر دوره زمانی به خوبی انتخاب شود و کارها زمان پردازش کوتاهی داشته باشند. اما در سایر موارد، ممکن است زمان انتظار در Round Robin بیشتر از FCFS باشد.

ح

نادرست، FCFS (First-Come, First-Served) یک الگوریتم غیر preemptive است، یعنی وقتی یک پردازه شروع به اجرا می‌کند، تا پایان اجرای آن ادامه می‌یابد و نیازی به preemption ندارد. اما Round Robin یک الگوریتم preemptive است. در Round Robin، هر پردازه برای مدت زمان مشخصی (time slice) اجرا می‌شود و پس از آن، اگر همچنان کاری برای انجام دادن داشته باشد، به انتهای صف انتظار منتقل می‌شود و پردازه بعدی شروع به اجرا می‌کند. این فرآیند نیازمند preemption است.

ط

نادرست، تأثیر الگوریتم‌های برنامه‌ریزی بر کارایی حافظه نهان به شدت به نوع برنامه‌ها و مدل دسترسی‌های حافظه آن‌ها بستگی دارد. در مواردی که برنامه‌ها به طور متناوب و با فاصله زمانی کوتاه اجرا شوند، ممکن است Round Robin باعث بهبود دسترسی به داده‌های حافظه نهان شود، زیرا این الگوریتم از ایجاد وقفه‌های طولانی در اجرای هر پردازه جلوگیری می‌کند. اما در سایر موارد، به خصوص زمانی که پردازه‌ها دارای دسترسی‌های حافظه متمرکز و طولانی هستند، FCFS ممکن است باعث کارایی بیشتر حافظه نهان شود.

ی

نادرست، وقتی یک ریسه درون یک پردازه file descriptor را می‌بندد، تنها برای آن پردازه بسته می‌شود، نه برای کل سیستم یا سایر پردازه‌ها. File descriptor ها در سطح پردازه مدیریت می‌شوند، نه در سطح سیستم. بنابراین، بسته شدن یک file descriptor توسط یک ریسه تنها برای همان پردازه‌ای که ریسه به آن تعلق دارد، تأثیر دارد.

ک

درست، در پردازه‌های چند ریسه‌ای، هر ریسه دارای پشته (stack) مخصوص به خود است. متغیرهایی که در پشته یک ریسه ذخیره می‌شوند، تنها توسط همان ریسه قابل دسترسی هستند و از دیگر ریسه‌ها مجزا هستند. بنابراین، دسترسی به این متغیرها thread-safe است، زیرا هیچ تداخلی بین ریسه‌ها در دسترسی به این متغیرها وجود ندارد.

ل

نادرست، اگرچه این الگوریتم می‌تواند به لحاظ نظری کارایی بالایی در کاهش زمان انتظار داشته باشد، اما بهینه‌ترین الگوریتم برای همه سناریوها و محیط‌های سیستم عامل نیست. عملکرد و کارایی یک الگوریتم زمان‌بندی به شدت به ماهیت بار کاری و خصوصیات سیستم وابسته است. همچنین، SRTF می‌تواند منجر به مشکلاتی مانند starvation برای پردازه‌های با زمان پردازش بلند شود.

پاسخ مسئله‌ی ۲.

آ

اتفاقاتی که باید ذخیره شوند:

مقادیر رجیسترها: شامل رجیسترهای محلی ریسه، مانند مقادیر پایینتر و بالاتر stack pointer، program counter و سایر رجیسترهای مربوط به وضعیت فعلی ریسه.

مقادیر مربوط به اجرای ریسه: این شامل مواردی مانند اولویت ریسه و وضعیت آماده به اجرا یا در انتظار است.

از آنجایی که هر دو ریسه به یک پردازه تعلق دارند، داده‌های مربوط به فضای کاربر پردازه (مانند فضای آدرس حافظه) مشترک هستند و نیازی به ذخیره و بازیابی مجدد آن‌ها در یک context switch داخلی نیست.

ب

مقادیر رجیسترها: مشابه حالت الف، مقادیر رجیسترهای هر ریسه باید ذخیره و بازیابی شوند.

فضای آدرس حافظه: از آنجایی که ریسه‌ها به پردازه‌های مختلف تعلق دارند، فضای آدرس حافظه هر پردازه (شامل بخش‌هایی مانند داده‌ها، کد و پشته) باید ذخیره و هنگام بازگشت به آن پردازه بازیابی شود.

اطلاعات مربوط به حافظه نهان و سایر منابع سیستم: این شامل اطلاعاتی است که برای حفظ وضعیت پردازه و بهینه‌سازی دسترسی‌های حافظه لازم است.