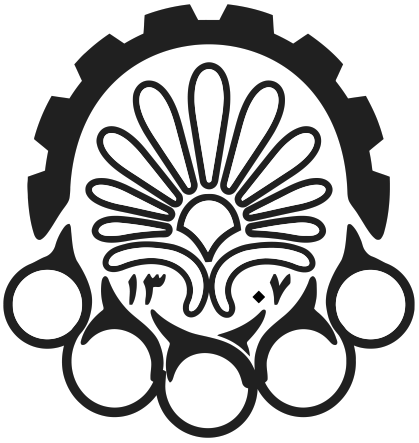
سیستم‌های عامل

دکتر زرندی



دانشگاه صنعتی امیرکبیر

(پلی تکنیک تهران)

دانشکده مهندسی کامپیوتر

رضا آدینه پور ۴۰۲۱۳۱۰۵۵

تمرین سری هفتم

۸ آذر ۱۴۰۳

سیستم‌های عامل

تمرین سری هفتم

رضا آدینه پور ۴۰۲۱۳۱۰۵۵

دانشکده مهندسی کامپیوتر

سوال اول

عبارات و اصطلاحات زیر را تعریف کنید:

۱. CPU Burst Time:

پاسخ

CPU Burst Time یعنی مدت زمانی که یک فرآیند (یک برنامه یا کار در حال اجرا) به طور پیوسته از CPU استفاده می‌کند. به عبارت دیگر، این زمان مشخص می‌کند که یک فرآیند چقدر زمان نیاز دارد تا کارهای خود را با پردازنده انجام دهد.
فرض شود یک برنامه در حال اجرا است. این برنامه ممکن است در زمان‌های مختلف به CPU نیاز داشته باشد تا محاسبات یا پردازش‌هایی انجام دهد. CPU Burst Time همان مدت زمانی است که این فرآیند به طور مداوم در حال استفاده از CPU است تا کار خود را انجام دهد. بعد از این مدت، ممکن است برنامه نیاز به انتظار برای I/O (مثلاً خواندن داده از دیسک یا شبکه) داشته باشد، و در این زمان دیگر پردازنده در اختیار برنامه نخواهد بود.

۲. Turnaround Time:

پاسخ

Turnaround Time: مجموع زمانی است که از زمان شروع یک فرآیند تا زمان اتمام آن طول می‌کشد. این زمان شامل:

(آ) زمان اجرای فرآیند (CPU Burst Time)

(ب) زمان انتظار برای منابع دیگر (مانند I/O یا دسترسی به پردازنده)

(ج) زمان ارسال و دریافت ورودی/خروجی

به طور کلی داریم:

$$\text{Time Turnaround} = T_{Begin} - T_{End}$$

۳. بن‌بست:

پاسخ

بن‌بست یا Deadlock یک وضعیت در سیستم‌های عامل است که در آن دو یا چند فرآیند یا thread به طوری به یکدیگر وابسته می‌شوند که هیچ کدام از آن‌ها قادر به ادامه اجرای خود نیستند. این وضعیت زمانی اتفاق می‌افتد که:

(آ) هر فرآیند یک یا چند منبع را در اختیار دارد.

(ب) هر فرآیند منتظر منبع دیگری است که توسط فرآیند دیگر نگهداری می‌شود.

در نتیجه، هیچ یک از فرآیندها نمی‌توانند ادامه یابند، زیرا هر کدام به منابعی نیاز دارند که در حال حاضر توسط دیگران قفل شده است.

۴. حالت امن:

پاسخ

حالت امن (Safe State) به حالتی اطلاق می‌شود که سیستم در آن قادر است به گونه‌ای منابع را تخصیص دهد که هیچ‌گاه به Deadlock منتهی نشود. به عبارت دیگر، در حالت امن، سیستم می‌تواند به راحتی منابع را بین فرآیندها تخصیص دهد بدون اینکه در هر مرحله‌ای وارد بن‌بست شود.
برای بررسی اینکه آیا سیستم در حالت امن است یا خیر، از الگوریتم‌های مانند الگوریتم Banker's Algorithm استفاده می‌شود، که بررسی می‌کند آیا می‌توان به نحوی منابع را تخصیص داد که همواره به فرآیندها اجازه داده شود تا به طور کامل به اتمام برسند.

سوال دوم

تصور کنید در یک سیستم ۵ فرآیند وجود دارد، که زمان ورود Arrival Time و زمان پردازش CPU Burst Time آن به‌صورت زیر می‌باشد.

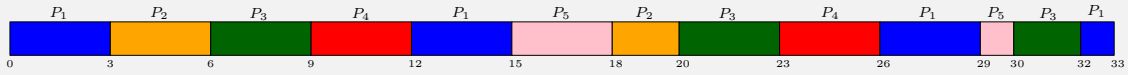
فرآیند	زمان ورود	مدت زمان پردازش
P_1	۰	۱۰
P_2	۱	۵
P_3	۲	۸
P_4	۳	۶
P_5	۴	۴

فرض کنید کوانتوم زمانی برابر با ۳ واحد زمانی است.

۱. نمودار گانت مربوط به این فرآیند ها را رسم کنید

پاسخ

با فرض Preemptive بودن، نمودار گانت به‌صورت زیر به‌دست می‌آید:



شکل ۱: گانت چارت

۲. زمان تکمیل (TimeCompletion) و زمان بازگشت (Turnaround Time) و زمان انتظار (Waiting Time) هر فرآیند را محاسبه کنید.

پاسخ

می‌دانیم:

$$TAT = CT - AT$$

$$WT = TAT - BT$$

بنابراین محاسبه می‌شود:

پردازه	زمان تکمیل (CT)	زمان بازگشت (TAT)	زمان انتظار (WT)
P_1	۳۳	۳۳	۲۳
P_2	۲۰	۱۹	۱۴
P_3	۳۲	۳۰	۲۲
P_4	۲۶	۲۳	۱۷
P_5	۳۰	۲۶	۲۲

۳. میانگین زمان انتظار (Average Waiting Time) و میانگین زمان بازگشت Turnaround Average Time را محاسبه کنید.

پاسخ

$$\text{Average Waiting Time} = \frac{(23 + 14 + 22 + 17 + 22)}{5} = \frac{98}{5} = 19.6$$

$$\text{Average Turnaround Time} = \frac{(33 + 19 + 30 + 23 + 26)}{5} = \frac{131}{5} = 26.2$$

سوال سوم

فرض کنید یک سیستم دارای ۶ فرایند $(P_0, P_1, P_2, P_3, P_4, P_5, P_6)$ و چهر نوع منبع (A, B, C, D) است که از هرکدام به ترتیب و درمجموع $(14, 10, 9, 12)$ موجود است. جدول زیر وضعیت فعلی تخصیص منابع را نشان می‌دهد.

D	C	B	A	Process
۲	۰	۱	۱	P_0
۰	۱	۱	۲	P_1
۳	۲	۱	۰	P_2
۲	۱	۰	۲	P_3
۱	۱	۲	۱	P_4
۰	۰	۲	۱	P_5

و جدول زیر بیشترین مقدار منابع مورد نیاز هر فرآیند را نشان می‌دهد:

D	C	B	A	Process
۳	۲	۱	۴	P_0
۷	۵	۳	۶	P_1
۹	۳	۵	۲	P_2
۴	۲	۲	۵	P_3
۵	۳	۳	۴	P_4
۶	۲	۵	۴	P_5

۱. آیا سیستم در حالت امن است؟

پاسخ

ابتدا ماتریس Need را محاسبه می‌کنیم. می‌دانیم که:

$$Need = Max - Alloc$$

پس ماتریس Need به‌صورت زیر محاسبه می‌شود:

D	C	B	A	Process
۱	۲	۰	۳	P_0
۷	۴	۲	۴	P_1
۶	۱	۴	۲	P_2
۲	۱	۲	۳	P_3
۴	۲	۱	۳	P_4
۶	۲	۳	۳	P_5

با به‌دست آوردن ماتریس Need و مقادیر Available که در صورت سوال داده‌شده است (۱۲، ۹، ۱۰، ۱۴) می‌توان گفت که سیستم درحالت امن است و همه فرآیند ها می‌توانند بدون مشکل اجرا شوند.

۲. اگر فرآیند P_1 درخواست $[1, 1, 2, 2]$ از منابع را ارسال کند، آیا این درخواست قابل قبول است؟

پاسخ

بله قابل قبول است. پس از این درخواست مقدار Available به‌صورت $(13, 9, 7, 10)$ می‌شود.

سوال چهارم

۱. به چه دلیلی الگوریتم SJF غیرقابل پیاده‌سازی در زمانبند کوتاه‌مدت است؟

پاسخ

الگوریتم SJF به رغم اینکه به طور نظری بهترین عملکرد را از نظر turnaround time دارد، در عمل غیرقابل پیاده‌سازی در زمان‌بند کوتاه‌مدت است به دلایل زیر:

(آ) نیاز به پیش‌بینی CPU Burst Time:
الگوریتم SJF برای انتخاب فرآیند بعدی که باید اجرا شود، به زمان پردازش باقی‌مانده هر فرآیند نیاز دارد. این زمان پردازش به طور دقیق باید پیش‌بینی شود تا بتوان فرآیندهایی با کمترین زمان پردازش را اولویت‌بندی کرد. در بسیاری از موارد، پیش‌بینی دقیق این زمان در زمان‌بند کوتاه‌مدت امکان‌پذیر نیست، زیرا نمی‌توان زمان دقیق پردازش هر فرآیند را قبل از شروع آن دانست.

(ب) اطلاعات مورد نیاز برای اجرای الگوریتم:
برای اعمال SJF به طور دقیق، سیستم باید از اطلاعات زمان پردازش فرآیندها به‌طور کامل آگاه باشد. این امر تنها در صورتی ممکن است که سیستم اطلاعات دقیقی از رفتار تمامی فرآیندها داشته باشد، که این اطلاعات معمولاً در زمان‌بند کوتاه‌مدت قابل دسترسی نیست. در سیستم‌های واقعی، زمان پردازش هر فرآیند ممکن است به صورت پویا تغییر کند، به‌ویژه در شرایطی که منابع دیگر (مثل ورودی/خروجی) همزمان در دسترس باشند.

(ج) زمان‌بندی در محیط‌های پیشرفته و multitasking:
در محیط‌های multitasking که در آن فرآیندهای متعدد به طور هم‌زمان در حال اجرا هستند و سیستم باید به طور مداوم منابع را بین فرآیندها تقسیم کند، پیاده‌سازی SJF بسیار پیچیده است. نیاز به محاسبه زمان پردازش باقی‌مانده و مرتب‌سازی پیوسته فرآیندها به منظور اولویت‌دهی، باعث ایجاد overhead قابل توجه در سیستم می‌شود و کارایی سیستم را تحت تأثیر قرار می‌دهد.

(د) دوری از اولویت‌دهی به فرآیندهای جدید:
الگوریتم SJF ممکن است باعث Starvation شود، به این معنا که اگر فرآیندی زمان پردازش طولانی داشته باشد، ممکن است همیشه بعد از فرآیندهایی با زمان پردازش کوتاه‌تر قرار گیرد و به این ترتیب هیچ‌گاه اجرا نشود. این مشکل در سیستم‌هایی که فرآیندهای جدید به طور مداوم وارد صف می‌شوند، جدی‌تر می‌شود.

۲. مزایا و معایب الگوریتم بانکداران چیست؟ توضیح دهید.

پاسخ

(آ) **مزایا:**

- **جلوگیری از Deadlock**: بزرگ‌ترین مزیت Banker's Algorithm این است که از ایجاد Deadlock جلوگیری می‌کند. با تجزیه و تحلیل وضعیت تخصیص منابع، سیستم می‌تواند اطمینان حاصل کند که فرآیندها هیچ‌گاه وارد وضعیت خطرناک نخواهند شد و بنابراین از Deadlock جلوگیری می‌شود.

پاسخ

- **اعتماد به تخصیص‌های امن:** الگوریتم به سیستم اجازه می‌دهد تا تخصیص منابع را فقط در صورتی انجام دهد که مطمئن باشد سیستم همچنان قادر به انجام کارهای بعدی بدون خطر Deadlock خواهد بود. این به سیستم کمک می‌کند تا منابع را به گونه‌ای تخصیص دهد که تمام فرآیندها قادر به تکمیل کار خود باشند.
- **انعطاف‌پذیری با تعداد فرآیندها و منابع:** این الگوریتم می‌تواند برای سیستم‌های با تعداد زیاد فرآیند و منابع به طور موثر استفاده شود. سیستم می‌تواند منابع را به طور پویا تخصیص دهد و همچنان وضعیت امن را حفظ کند.
- **پیش‌بینی در تخصیص منابع:** الگوریتم Banker's به مدیر سیستم این امکان را می‌دهد که پیش‌بینی کند آیا تخصیص منابع در آینده باعث ایجاد مشکلاتی مانند Deadlock خواهد شد یا نه. این ویژگی به بهینه‌سازی استفاده از منابع کمک می‌کند.

(ب) **معایب:**

- **پیچیدگی محاسباتی بالا:** یکی از معایب اصلی الگوریتم Banker's این است که محاسبات آن پیچیده است. برای هر تخصیص منابع، باید وضعیت سیستم بررسی شود تا معلوم شود آیا وضعیت سیستم امن است یا نه. این بررسی‌ها شامل محاسبات زیادی هستند که به ویژه در سیستم‌هایی با تعداد زیادی فرآیند و منابع می‌تواند زمان‌بر باشد.
- **نیاز به اطلاعات دقیق:** الگوریتم Banker's به اطلاعات دقیق و به روز در مورد نیازهای منابع فرآیندها نیاز دارد. این شامل اطلاعاتی درباره تعداد منابع مورد نیاز در هر لحظه و همچنین میزان منابع آزاد در سیستم است. جمع‌آوری و نگهداری این اطلاعات ممکن است پیچیده و زمان‌بر باشد.
- **Overhead بالای پردازشی:** با توجه به اینکه الگوریتم باید وضعیت سیستم را به طور مداوم بررسی کند تا از Deadlock جلوگیری کند، در سیستم‌های با بار کاری سنگین، این بررسی‌ها می‌تواند باعث افزایش overhead شود که کارایی سیستم را کاهش می‌دهد.
- **ممکن است باعث کاهش کارایی شود:** برای اطمینان از وضعیت امن، گاهی ممکن است تخصیص منابع به برخی فرآیندها به تأخیر بیفتد، حتی اگر این تخصیص‌ها ضروری نباشند. این می‌تواند منجر به کاهش کارایی و عدم استفاده بهینه از منابع شود.
- **ممکن است منجر به Starvation شود:** اگر یک فرآیند همیشه در حال انتظار برای تخصیص منابع باشد (چرا که همیشه تخصیص آن باعث وضعیت خطرناک می‌شود)، ممکن است starvation رخ دهد. به این معنی که یک فرآیند به دلیل تخصیص منابع به فرآیندهای دیگر، هرگز منابع مورد نیاز خود را دریافت نکند.
- **نیاز به پیش‌بینی نیازهای آینده:** الگوریتم Banker's نیاز به پیش‌بینی منابع مورد نیاز فرآیندها در آینده دارد. این امر در بسیاری از موارد عملی نیست، زیرا بسیاری از فرآیندها نیازهای خود را به طور پویا و در طول زمان تغییر می‌دهند.

۳. روش‌های بازیابی از بن‌بست چیست؟ این روش‌ها را شرح دهید و با یکدیگر مقایسه کنید.

پاسخ

۱. Process Termination: در این روش، سیستم یکی از فرآیندهای درگیر در Deadlock را خاتمه می‌دهد تا منابع آن آزاد شوند و Deadlock رفع شود. این روش می‌تواند به دو صورت انجام شود:

(آ) **خاتمه فرآیندها به ترتیب انتخابی**: فرآیندهایی که کمترین اولویت دارند یا کمترین هزینه را برای سیستم دارند خاتمه داده می‌شوند.

(ب) **خاتمه تمامی فرآیندهای درگیر در Deadlock**: تمام فرآیندهایی که در وضعیت Deadlock هستند خاتمه می‌یابند.

مزایا:

- ساده و سریع است.
- برطرف کردن فوری Deadlock بدون نیاز به تغییرات پیچیده در وضعیت سیستم.

معایب:

- ممکن است باعث از دست رفتن داده‌ها و کارهای نیمه‌تمام فرآیندها شود.
- در صورت انتخاب اشتباه فرآیند، ممکن است سیستم کارآیی خود را از دست بدهد.
- ممکن است هزینه زیادی از نظر زمان و منابع داشته باشد.

۲. Resource Preemption: در این روش، منابع از فرآیندهایی که در وضعیت Deadlock قرار دارند بازپس‌گیری می‌شود و به فرآیندهای دیگر اختصاص می‌یابد. این روش معمولاً برای آزادسازی منابع از فرآیندهایی که کمترین نیاز را به آن‌ها دارند استفاده می‌شود.

مزایا:

- فرآیندها می‌توانند به اجرای خود ادامه دهند و Deadlock رفع می‌شود.
- در صورتی که منابع به درستی بازپس‌گیری شوند، فرآیندهای دیگر می‌توانند از آن‌ها بهره‌برداری کنند.

معایب:

- باعث Starvation می‌شود، چون ممکن است منابع به فرآیندهای دیگر اختصاص یابد و فرآیندهای درگیر در Deadlock همیشه منتظر بمانند.
- ممکن است به داده‌های مهم فرآیندها آسیب برسد.
- هزینه‌های زیادی برای مدیریت و زمان‌بندی منابع دارد.

۳. Process Rollback: در این روش، سیستم فرآیندهایی را که در Deadlock قرار دارند به وضعیت قبلی خود بازمی‌گرداند. این کار معمولاً با استفاده از checkpointing (نقطه‌چک) انجام می‌شود، که در آن وضعیت‌ها و داده‌های مربوط به فرآیندها ذخیره می‌شوند. زمانی که Deadlock شناسایی می‌شود، سیستم فرآیند را به آخرین وضعیت ذخیره‌شده بازمی‌گرداند.

مزایا:

- اطلاعات از دست نمی‌رود و فرآیندها به صورت کامل بازیابی می‌شوند.
- در صورت استفاده صحیح از نقاط ذخیره‌سازی (checkpoints)، سیستم می‌تواند بدون از دست رفتن داده‌ها بازیابی شود.

پاسخ

معایب:

- به هزینه ذخیره‌سازی و مدیریت نیاز دارد.
- ممکن است زمان‌بری باشد، به خصوص اگر فرآیندها حجم داده‌های زیادی را ذخیره کنند.
- Rollback ممکن است به Deadlock جدیدی منجر شود.

۱. Killing the Resources: در این روش، به جای خاتمه دادن به فرآیندها، سیستم تمام منابعی که باعث Deadlock شده‌اند را آزاد می‌کند. به این معنی که منابع درگیر در Deadlock به طور کلی آزاد شده و به سایر فرآیندها اختصاص داده می‌شود.

مزایا:

- ساده و سریع است.
- موجب آزادسازی فوری منابع می‌شود.

معایب:

- موجب اتلاف منابع می‌شود.
- ممکن است تغییرات زیادی در وضعیت سیستم ایجاد کند.

و درنهایت در جدول زیر مقایسه‌ای از این روش‌ها را ارائه می‌دهم:

روش	مزایا	معایب
Process Termination	ساده و سریع، حل فوری Deadlock	ممکن است داده‌ها از دست بروند، هزینه زیاد در انتخاب فرآیند
Resource Preemption	فرآیندها می‌توانند ادامه دهند، بازگشت منابع	ممکن است باعث Starvation و آسیب به داده‌ها شود
Process Rollback	اطلاعات از دست نمی‌رود، بازیابی دقیق	هزینه ذخیره‌سازی بالا، ممکن است به Deadlock جدید منجر شود
Killing the Resources	آزادسازی فوری منابع	موجب اتلاف منابع، تغییرات زیاد در وضعیت سیستم

جدول ۱: مقایسه روش‌های بازیابی از Deadlock