سیستم‌های عامل - دکتر ابراهیمی‌مقدم

امیرحسین منصوری - ۹۹۲۴۳۰۶۹

تمرین سری ششم

سوال ۱

این که یک پروسه با یک منبع چه کاری می‌کند اهمیتی ندارد؛ صرف در اختیار داشتن منبع می‌تواند باعث Deadlock شود. بنابراین برای انتخاب پروسهٔ موردنظر برای نجات پیدا کردن از وضعیت Deadlock، صرفا به وضعیت اختصاص منابع نگاه می‌کنیم. پس مورد سوم تاثیری ندارد.

سوال ۲

مورد اول درست است. صورت درست گزینه‌های دیگر به شکل زیر است:

- اگر در وضعیت unsafe باشیم، *ممکن است* Deadlock رخ دهد.
- الگوریتم Banker اتفاقا برای سیستمی با چندین منابع که از هر کدام چندین نوع وجود دارد طراحی شده است.
- در هنگام بسته شدن پروسه برای شکستن وضعیت Deadlock، همهٔ منابع آن پروسه که گرفته شده بودند آزاد می‌شوند.

سوال ۳

- جمله درست است.
- وجود دور در گراف، اگر از هر منبع چندین نمونه داشته باشیم، *می‌تواند* نشانهٔ Deadlock باشد.
- جمله درست است.
- جمله درست است؛ چون شرط Circular wait که برای وقوع Deadlock لازم است، نقض می‌شود.
- جمله درست است.
- اگر از هر منبع فقط یک نمونه داشته باشیم، و دور داشته باشیم، حتما Deadlock رخ داده است.

سوال ۴

در Deadlock prevention، ما شرایط و محدودیت‌هایی را بر درخواست‌های منابع اعمال می‌کنیم تا بتوانیم یکی از شرط‌های ۴گانه وقوع Deadlock را نقض کنیم و از این طریق جلوی آن را بگیریم.

در Deadlock avoidance سعی می‌کنیم در هر بار درخواست منابع، بررسی کنیم آیا این درخواست ما را به وضعیتی که در آن *احتمال* وقوع Deadlock وجود دارد (یا به عبارتی Unsafe state) می‌برد یا نه؛ و در این صورت آن درخواست را رد می‌کنیم. از این نظر، در Deadlock avoidance کمی محتاط‌تر هستیم.

سوال ۵

این الگوریتم برای تشخیص Safe بودن وضعیت اختصاص منابع است. در این الگوریتم، در هر مرحله سعی می‌کنیم یکی از تسک‌هایی که امکان فراهم‌کردن منابعش وجود دارد (با توجه به منابع آزاد) را انتخاب کنیم، و سپس آن را تمام شده فرض کنیم و منابعش را به منابع آزاد اضافه کنیم. این کار را تا جایی ادامه می‌دهیم که همهٔ تسک‌ها تمام شوند. اگر در هر مرحله نتوانیم تسکی را انتخاب کنیم، یعنی سیستم در وضعیت Unsafe بوده است؛ در غیر این صورت در وضعیت Safe قرار داشته‌ایم.

یک ایراد این الگوریتم این است که باید حداکثر منابع استفاده شده هر تسک را بدانیم؛ که همیشه ممکن نیست. ایراد دیگر این است که این الگوریتم سخت‌گیرانه عمل می‌کند؛ به این معنی که یک وضعیت Unsafe، الزاما به Deadlock منجر نمی‌شود؛ اما توسط این الگوریتم به عنوان یک وضعیت نامطلوب درنظر گرفته می‌شود.

سوال ۶

یک روش این است که روی انواع منبع، ترتیبی تعریف کنیم (مثلا به هر کدام یک عدد یکتا بدهیم)، و قاعده‌ای را تعریف کنیم که طبق آن، تردها حتما منابع را به ترتیب مشخص‌شده درخواست کنند. مثلا اگر یک ترد می‌خواهد دو نوع منبع با شماره‌های ۳ و ۵ را درخواست کند، حتما باید منبع با شماره ۳ را اول درخواست کند و سپس سراغ درخواست منبع شماره ۵ برود.

سوال ۷

الف)

جمله صحیح است. کلاً سه حالت برای نشستن فیلسوف‌ها وجود دارد:

- یک فیلسوف از یک نوع، و چهار فیلسوف از نوع دیگر
- دو فیلسوف از یک نوع، سه فیلسوف از نوع دیگر، به طوری که فیلسوف‌های از یک نوع کنار هم بشینند.
- دو فیلسوف از یک نوع، سه فیلسوف از نوع دیگر، به طوری که مانند حالت بالا نباشد.

در همهٔ این حالات، حتی اگر فیلسوف‌ها با هم اولین چنگال را بردارند، باز هم بالاخره یک فیلسوف موفق به برداشتن هر دو چنگال می‌شود. بنابراین Deadlock رخ نمی‌دهد.

ب)

می‌توان بررسی کرد که این حالت چنگال برداشتن فیلسوف‌ها (دو فیلسوف از یک نوع و سه فیلسوف از نوع دیگر)، فقط دو حالت مختلف نشستن می‌تواند ایجاد کند. در هر کدام از این دو حالت، حتی اگر همهٔ فیلسوف‌ها با هم اولین چنگال را بردارند، باز هم Deadlock رخ نمی‌دهد؛ چون حداقل یک فیلسوف موفق به برداشتن هر دو چنگال خواهد شد.

ج)

در قسمت ب، حالت‌هایی را داشتیم که دو فیلسوف با نوع چنگال‌برداشتن یکسان کنار هم بودند. پس این موضوع نیز باعث Deadlock نمی‌شود.

د)

در قسمت ب نیز بدترین حالتی که داشتیم، همزمان برداشتن چنگال‌ها توسط فیلسوف‌ها بود. بنابراین این موضوع هم باعث Deadlock نمی‌شود.

سوال ۸

الف)

در این وضعیت Deadlock نداریم.

- R_2 در اختیار T_2 قرار دارد و این تسک به پایان می‌رسد.
- R_2 را به T_3 می‌دهیم و این تسک نیز تمام می‌شود.
- R_1 و R_2 را به T_1 می‌دهیم و تمام تسک‌ها تمام می‌شوند.

ب)

T_1 و T_3 در وضعیت Deadlock قرار دارند. حلقهٔ $< T_1, R_3, T_3, R_1, T_1 >$ و این که از هر منبع یک نمونه داریم، این موضوع را نشان می‌دهد.

ج)

در این وضعیت Deadlock نداریم.

- T_2 منابع R_1 و R_2 را در اختیار دارد؛ بنابراین کارش تمام می‌شود و یک نمونه از هر دو منبع آزاد می‌شود.
- T_3 نیز منابع R_1 و R_2 را در اختیار دارد؛ بنابراین کارش تمام می‌شود و هر دو منبع کامل آزاد می‌شوند.
- به T_1 منابع R_1 و R_2 را می‌دهیم و کار این تسک هم تمام می‌شود.

د)

حلقهٔ زیر وجود دارد:

$$< T_1, R_2, T_3, R_1, T_2, R_2, T_4, R_1 >$$

می‌توان ادعا کرد که چون در این حلقه، هر منبع دوبار آمده و از هر منبع نیز دو نمونه داریم، بنابراین Deadlock رخ داده است.

سوال ۹

- P_3 همه منابعش مورد نیازش را دارد. پس این پروسه انجام می‌شود و کارش تمام می‌شود. همچنین یک نمونه از هر منبع R_2 و R_3 نیز آزاد می‌شود.
- P_2 حالا می‌تواند R_3 را در اختیار بگیرد و کارش را تمام کند. همچنین R_2 کامل آزاد می‌شود و از R_3 یک نمونه آزاد باقی می‌ماند.
- حالا P_0 نیز می‌تواند R_2 را در اختیار بگیرد و اجرا شود. بعد از این، همهٔ R_2ها و یک نمونه از R_1 آزاد می‌شود.
- در نهایت، P_0 اجرا می‌شود.

بنابراین $< P_3, P_2, P_0, P_1 >$ یک Safe Sequence خواهد بود.

سوال ۱۰

الف)

جدول Need بعد از قبول درخواست به صورت زیر خواهد بود.

Process	A	B	C
P0	5	4	4
P1	0	1	1
P2	3	1	3
P3	9	5	1
P4	9	7	7

همچنین $\text{Available} =< 1, 0, 2 >$.
هیچ پروسه‌ای قابل انجام نیست؛ پس قبول کردن این درخواست، ما را به Unsafe state می‌برد.

ب)

جدول Need بعد از قبول درخواست به صورت زیر خواهد بود.

Process	A	B	C
P0	7	5	4
P1	0	1	1
P2	3	1	3
P3	9	5	1
P4	8	7	6

همچنین $\text{Available} =< 2, 1, 1 >$.
در این حالت، فقط پروسهٔ P_1 قابل انجام است و بعد از انجام آن، $\text{Available} =< 5, 1, 2 >$. می‌توان دید که در این حالت، هیچ پروسهٔ دیگری قابل انجام نیست. پس قبول این درخواست، ما را به Unsafe state می‌برد.

ج)

آرایهٔ Need پس از قبول این درخواست به صورت

$$< 3, 3, 4, 1, 3 >$$

خواهد بود. همچنین $\text{Available} = 10 - (2 + 1 + 2 + 1 + 1) - 2 = 1$.
با اجرای P_3، خواهیم داشت $\text{Available} = 2$؛ و دیگر نمی‌توان هیچ پروسهٔ دیگری را اجرا کرد. بنابراین این درخواست ما را به Unsafe state خواهد برد.