جلسه هفتم

روش های اداره بن بست(Dead lock handling)

4 روش برخورد با بن بست وجود دارد.

1- **پیشگیری از بن بست:** برای پیشگیری از بن بست می بایست کاری کرد که یکی از شرایط چهارگانه وقوع بن بست نقض شود. بدین ترتیب هرگز بن بست رخ نخواهد داد.

- **نقض انحصار متقابل :** اگر از منابعی به صورت اشتراکی استفاده کنیم بن بست هیچگاه رخ نمی دهد، نمونه آن فایل های فقط خواندنی هستند که می توانند همزمان توسط چندین پردازش استفاده شوند. البته این روش برای یک سری از منابع که ذاتا ماهیت غیر اشتراکی دارند قابل استفاده نیست مثل چاپگر. البته با ***spool*** کردن خروجی چاپگر چندین پروسس می توانند در یک زمان خروجی های خود را تولید کنند ولی تمام دستگاه ها را نمی توان ***spool*** کرد مثل جدول پروسس.
- **نقضِ گرفتن و منتظر ماندن :** برای نقض این شرط، باید مانع از ایجاد موقعیتی شد که در آن یک پردازه، منبعی را در اختیار گیرد و تقاضای منبع دیگری را نماید. رسیدن به این هدف با دو روش امکان پذیر است.

A. همه منابع مورد نیاز پردازه ها در ابتدای شروع اجرای پردازه اگر در دسترس باشند به آن اختصاص داده شوند وگرنه اختصاص داده نشوند به عبارتی یا همه منابع اختصاص داده شوند یا هیچکدام اختصاص داده نشوند. به عنوان مثال پردازه ای که می بایست داده ها را از نوار مغناطیس خوانده سپس آنها را بر روی دیسکی کپی کرده و آنها را مرتب کرده و چاپ کند، می بایست در ابتدا چاپگر را که در انتهای کارش نیاز دارد به دست آورد ، پس از چاپگر به درستی استفاده نشده است ، زیرا در همین حین که چاپگر بدون کار در اختیار این پردازه بوده ، پردازه دیگری می توانست از این استفاده کند. پس عیب این پروتکل کاهش بهره وری سیستم است.

B. هر پردازه ای که یک سری منابع را در اختیار دارد، و طلب منابع دیگری می کند می بایست منابع در اختیارش را آزاد سازد، سپس تمام منابع را با هم تحویل بگیرد، مشکل این روش قحط زدگی (گرسنگی) است، یعنی پردازه ای که منابع متعددی نیاز دارد بایستی به طور نامعین در انتظار باشد چرا که به احتمال زیاد یکی از منابع مورد نیازش همواره توسط پردزاش دیگری استفاده شده است.

- **نقض عدم پس گرفتن:** جهت تحقق این شرط دو راه حل وجود دارد.

A. یکی این است که اگر پردازه ای درخواست منابعی را داد که آن منابع آزاد نباشند،تمام منابع در اختیار پردازه ی درخواست کننده پس گرفته شوند.

B. راه حل دیگر آن است که بررسی شود که آیا پردازه های منتظر تمام منابع مورد نیاز پردازه درخواست کننده را دارند یا نه، که اگر داشته باشند منابع از پردازه های منتظر گرفته شده و به پردازه درخواست کننده انتساب یابد و اگرپردازه یا پردازه های منتظر، منابع موردنیاز پردازه ی درخواست کننده را نداشته باشند، خود پردازه ی درخواست کننده باید منتظر بماند وممکن است در حین انتظار منابع در اختیارش نیز از وی گرفته شوند وبه پردازه های دیگری انتساب یابند.

- **نقض انتظار چرخشی:**

جهت نقض انتظار چرخشی می بایست منابع را شماره گذاری کرد به طوری که هر پردازه بتواند منابع را در جهت صعودی شماره هایشان درخواست کند. به عنوان مثال اگر پردازه ای منبع شماره 3 را در اختیار داشته باشد، منبع شماره ی 1 را نمی تواند درخواست بکند، ولی می تواند منبع شماره ی 5 را درخواست کند. در این درخواست برای یک منبع می توان چندین نمونه از آن منبع را طلب کرد. شماره گذاری منابع می بایست بر اساس ترتیب نیاز به منابع صورت گیرد.به عنوان مثال، در مثال قبل می بایست شماره ی نوار مغناطیسی کمتر از شماره ی دیسک و شماره ی دیسک کمتر ازشماره ی چاپگر باشد. جهت اثبات اینکه روش گفته شده انتظار چرخشی را نقض می کند از برهان خلف استفاده می کنیم. فرض کنیم با اعمال این

شرط باز هم انتظار چرخشی داشته باشیم. در این صورت اگر شماره منبع در اختیار پردازه ی p_{i-1} ، $f(R_{i-1})$ باشد،این پردازه در صورتی منتظر P_i است که $f(R_{i-1}) < f(R_i)$ با شد و با تعمیم این رابطه به رابطه ی می رسیم. که می دانیم

$$f(R_0) < f(R_1) < ... < f(R_n) < f(R_0) \qquad f(R_n) < f(R_0)$$

نیست و برهان خلف اثبات می شود.

عیب روش پیشگیری از بن بست، بهره وری پایین منابع و کاهش توان عملیاتی سیستم می باشد .

2- اجتناب از بن بست:

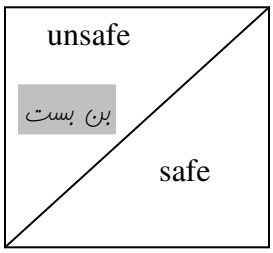
در این روش با توجه به اطلاعاتی نظیر حداکثر نیاز پردازه ها به منابع و منابع تخصیص یافته به پردازه ها و موجودی، وقتی پردازه ای درخواست یک سری منابع را داشته باشد، اگر منابع موجود باشند، بررسی می کنیم که آیا با اجابت این درخواست سیستم به حالت امن میرود یا نه. اگر سیستم به حالت امن برود درخواست تخصیص داده می شود و گرنه از درخواست اجتناب می شود .

حالت امن حالتی است که پردازه ها میتوانند به ترتیب خاص منابع مورد نیازشان را گرفته و اجرایشان را با موفقیت پشت سر بگذرانند.

تعریف دیگر: حالتی است که در آن یک ترتیب امن(Safe sequence) از پردازه ها وجود داشته باشد .

تعریف دیگر: اگر سیستم بتواند منابع مورد درخواست را به ترتیبی تخصیص دهد که از بروز بن بست اجتناب شود گوئیم آن سیستم د رحالت امن است

Safe sequence.(ترتیب امن) ترتیبی است از پردازه ها که می توانند با ترتیب معینی از تحویل گرفتن منابع،اجرایشان را خاتمه دهند.



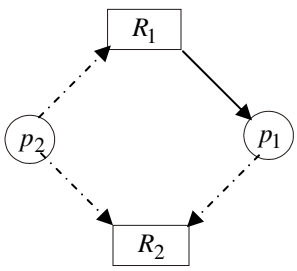
برای اجتناب از بن بست دو راه وجود دارد:

1- از هر منبع فقط یک نمونه وجود داشته باشد: از گرافی شبیه گراف تخصیص منابع استفاده می شود.

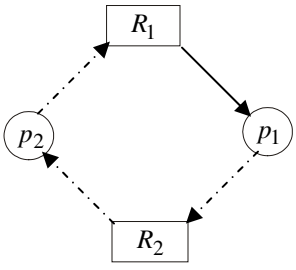
2- از هر منبع بیش از یک نمونه وجود داشته باشد. از الگوریتم بانکداران استفاده می شود.

گراف تخصیص منبع :این گراف شبیه گراف تخصیص منابع است که علاوه بر یال های انتظار وتخصیص شامل یک یال ادعا (claim eddge) می باشد که با نقطه چین نمایش داده می شود.به طوریکه p_i ⇢ R_i بدین معناست که ممکن است در آینده پردازه ی p_i درخواست منبع R_i را داشته باشد.هریال تخصیص پس از پایان تخصیص تبدیل به یال ادعا می شود.

با توجه به شکل اگر p_2 در لحظه ی فعلی درخواست R_2 را داشته باشد، این درخواست به وی اعطا نمی شود، زیرا سیستم به حالت نا امن می رود به طوری‌که ممکن است در آینده P_2 طلب R_1 و P_1 طلب R_2 را داشته باشد، که بن بست رخ می دهد



حالت ناامنی به این شکل ایجاد می شود



اگردر گراف بالایی P_1 درخواست R_2 را داشته باشد،آیا این درخواست اجابت شود یا خیر؟

بله،اجابت می شود، زیرا در آینده ایجاد حلقه غیر ممکن است.

الگوریتم بانکداران: این الگوریتم اولین بار توسط دیجسترا ارائه شد ، و به نام الگوریتم بانکدار معروف گردید، چرا که این الگوریتم شبیه، رفتار یک بانکدار شهر کوچک با مشتریانش طراحی شده است، یک بانکدار هرگز تمام سرمایه خودش را به مشتریان تخصیص نمی دهد و طوری عمل می کنند که بتواند کلیه نیاز های مشتریانش را بر آورده کند.

ساختمان داده های مورد نیاز:(n تعداد پردازه ها وm تعداد منابع)

1. $Max_{n\times m}$

- $Max[i][j]=k$ بدین معناست که پردازه p_i ، جهت اجرایش به k نمونه از منبع R_j نیاز دارد

2. $Alocation_{n\times m}$

- $Alocation[i][j]=k$ به این معناست که در حال حاضر k نمونه از منبع R_j در اختیار پردازه p_i ، می باشد.

3. $Need_{n\times m}$

- $Need[i][j]=k$ به این معناست که پردازه p_i جهت ادامه کارش به k نمونه از منبع R_j نیاز دارد.

- بدیهی است که $Need[i][j]=Max[i][j]-Alocation[i][j]$ می باشد
- بردار $Available$ (در دسترس) به طول m تعداد منابع آزاد از هر نوع را نشان می دهد. مثلا $Available[j]=k$ باشد یعنی از منبع نوع R_j به تعداد k نمونه در دسترس وجود دارد.

اگر $Request_i$ بردار درخواست منابع ، پردازه p_i باشد الگوریتم به صورت زیر نوشته می شود.

1- اگر $Need_i < \text{Request}_i$ (یعنی سطر مربوط به پردازه p_i در ماتریس) باشد،این درخواست غیر قانونی است ، و بررسی نمی شود(یعنی پایان الگوریتم)

- دو بردار x و y را به طول n در نظر بگیرید. می گوئیم $x \le y$ اگر و فقط اگر $x[i] \le y[i]$ باشد به ازاء $i=1,2,\cdots,n$. مثلا

$(0,4,2,3) \le (2,8,3,5)$

2- اگر $Request_i > Available$ باشدبدین معناست که منابع کافی جهت تخصیص به پردازه p_i ، وجود ندارد، و p_i می بایست منتظر بماند.

3- وانمود می کنیم که منابع مورد درخواست پردازه p_i ، به وی اعطاء می شود، بنابراین ساختمان داده ها به صورت زیر تغییر میی یابند.

$Alocation_i += Request_i$
$Need -= Request_i$
$Available -= Request_i$

4- فراخوانی الگوریتم امنیت: اگر حالت سیستم ناامن باشد ساختمان داده های تغییر یافته در قسمت 3 به حالت اول بر گردانده می شود(اجتناب از بن بست)

5- پایان

☹ مرتبه اجرای الگوریتم بانکدار $m\times n^2$ می باشد(n تعداد پردازش ها و m تعداد انواع منابع است)

الگوریتم امنیت

استفاده از بردار های work به طول m (متناظر با منابع) و Finish به طول n (متناظر با پردازه ها)

1- work=Available و برای Finish[i]=False : i=1…n

2- به ازای i=1,…,n (به ازای تمام مقایر پردازه ها) مقدار i را چنان بیابید که، Finish[i]=Fals و $Need_i \le work$ اگر چنین i ای پیدا نشد برو به مرحله ی 4

(در این مرحله به دنبال پردازه ای می گردیم که به پایان نرسیده باشد و با منابع موجود بتوان اجرایش را به پایان رساند)

3- زمانی به مرحله سه می آییم که پردازه ای پیدا شود که اجرایش تمام نشده و با استفاده از منابع موجود می تواند نیاز هایش را بر آورده کرده و اجرایش را به اتمام برساند بنابراین پس از اتمام اجرایش منابع در اختیارش را آزاد می سازد. $work+=Alocation$, $finish[i]=true$

4- اگر برای تمام i ها (i=1…n):Finish[i]=true باشد ، بدین معناست که تمام پردازه ها می توانند با منابع موجود اجرایشان را با موفقیت پشت سر بگذرانند و سیستم در حالت امن قرار دارد ولی اگر حداقل یک i پیدا شود که Finish[i]=false باشد بدین معناست که با منابع موجود این پر دازه نمی تواند اجرایش را به اتمام برساند و سیستم در حالت ناامن قرار دارد.

مثال. فرض کنید سیستمی با چهار نوع منبع R_1 , R_2 , R_3 , R_4 که هر کدام به ترتیب دارای 8 , 5 , 9 , 7 نمونه می باشند، موجود باشد اگر در این سیستم 5 پردازه ی p_1 الی p_5 را داشته باشیم، با توجه به ماتریس های max و *Alocation* تشخیص دهید سیستم در حالت امن است یا نه؟

process	R_1	R_2	R_3	R_4
p_1	2	0	1	1
p_2	0	1	2	1
p_3	4	0	0	3
p_4	0	2	1	0
p_5	1	0	3	3
sum	7	3	7	5

Alocation

process	R_1	R_2	R_3	R_4
p_1	3	2	1	4
p_2	0	2	5	2
p_3	5	1	0	5
p_4	1	5	3	0
p_5	3	0	3	3

max

حل. ابتدا ماتریس *Need* را می سازیم با توجه به $Need = Max - Alocation$ خواهیم داشت .

process	R_1	R_2	R_3	R_4
p_1	1	2	0	3
p_2	0	1	3	1
p_3	1	1	0	2
p_4	1	3	2	0
p_5	2	0	0	3

Need

حال بردار *Avalable* را با توجه به فرمول $Avalable =$ موجودی اولیه $-$ *sum* ایجاد می کنیم و بر دار *finish* و *work* نیز در ابتدا به شکل زیر می باشند.

	R_1	R_2	R_3	R_4
Available	1	2	2	2

	p_1	p_2	p_3	p_4	p_5
finish	*f*	*f*	*f*	*f*	*f*

	R_1	R_2	R_3	R_4
work	1	2	2	2

پردازه هائی را که *false* هستند با *work* مقایسه می کنیم.(مقایسه سطر های ماتریس *Need* با *work*)، که در اولین مقایسه ملاحظه می شود که فقط پردازه p_3 کوچکتر است و می تواند اجرایش را با موفقیت انجام دهد و پس از اجرا منابع را بر گرداند، کار را به همین ترتیب ادامه می دهیم.

	R_1	R_2	R_3	R_4
work	5	2	2	5

	p_1	p_2	p_3	p_4	p_5
finish	*f*	*f*	*T*	*f*	*f*

	R_1	R_2	R_3	R_4
work	7	2	3	6

	p_1	p_2	p_3	p_4	p_5
finish	*T*	*f*	*T*	*f*	*f*

	R_1	R_2	R_3	R_4
work	7	3	5	7

	p_1	p_2	p_3	p_4	p_5
finish	*T*	*T*	*T*	*f*	*f*

	R_1	R_2	R_3	R_4
work	7	5	6	7

	p_1	p_2	p_3	p_4	p_5
finish	*T*	*T*	*T*	*T*	*f*

	R_1	R_2	R_3	R_4
work	8	5	9	7

	p_1	p_2	p_3	p_4	p_5
finish	*T*	*T*	*T*	*T*	*T*

ملاحظه می شود که سیستم در حالت امن است وترتیب اجرا به صورت p_3, p_1, p_2, p_4, p_5 می باشد

در همین سئوال پردازه p_3 درخواست یک نمونه از منبع R_3 را داشته باشد آیا این درخواست پاسخ داده شود یا خیر؟

مثال.سیستمی با پنج فرایند p_0 الی p_4 و سه نوع منبع C, B, A را در نظر بگیرید منبع نوع A دارای 10 نمونه، منبع نوع B دارای 5 نمونه و منبع نوع C دارای 8 نمونه می باشد، فرض کنید در زمان t_0 وضعیت سیستم به صورت زیر باشد

process	*A*	*B*	*C*
p_0	7	5	3
p_1	3	2	2
p_2	9	0	2
p_3	2	2	2
p_4	4	3	3

max

process	*A*	*B*	*C*
p_0	0	1	0
p_1	2	0	0
p_2	3	0	2
p_3	2	1	1
p_4	0	0	2

Allocation

الف. ماتریس *Need* را بیابید

ب. آیا در لحظه فعلی سیستم در حالت امن است یا نه؟ اگر در حالت امن است توالی امن را بیابید

ج. فرض کنید در لحظه t_0 فرایند p_1 یک نمونه از منبع A و دو نمونه دیگر از منبع C درخواست کند آیا این درخواست فورا پاسخ داده شود یا خیر.

د. اگر پردازه p_4 در خواست سه نمونه از منبع A و سه نمونه دیگر از منبع B را درخواست کند، آیا این درخواست می بایست بر آورده شود یا خیر.

process	A	B	C
p_0	7	4	3
p_1	1	2	2
p_2	6	0	0
p_3	0	1	1
p_4	4	3	1

Need

حل.

الف. با توجه به $Need = \max - Allocation$ داریم

ب. ابتدا بردار *Available* را می سازیم

A (آزاد) منابع موجود $= 10-(2+3+2) = 3$ $\quad B$ (آزاد) منابع موجود $= 5-(1+1) = 3$ $\quad C$ (آزاد) منابع موجود $= 7-(2+1+2) = 2$

پس بردار منابع در دسترس(*Available*) به صورت (3,3,2) می باشد که با توجه به این بردار مسئله را حل می کنیم.

منابع آزاد $(3,3,2) \xrightarrow{p_1} (5,3,2) \xrightarrow{p_3} (7,4,3) \xrightarrow{p_4} (7,4,5) \xrightarrow{p_2} (10,4,7) \xrightarrow{p_0} (10,5,7)$

همانطور که مشاهده می شود ترتیب اجراء $< p_1, p_3, p_4, p_2, p_0 >$ یک ترتیب امن است . لذا سیستم در حالت امن قرار دارد

نکته:اگر در یک لحظه چندین پروسس شرط لازم را برای اجرا شدن داشته باشند، اهمیتی ندارد که کدام یک جهت تخصیص منابع انتخاب شود چرا که در هر حال آن پروسس پس از تخصیص منابع مورد نیاز و تمام شدن،کل منابع خود را آزاد می کند.

ب. اگر درخواست فوق را برآورده سازیم ماتریس های *Allocation* و *Need* به صورت زیر در خواهند آمد

process	A	B	C
p_0	0	1	0
p_1	❸	0	❷
p_2	3	0	2
p_3	2	1	1
p_4	0	0	2

Allocation

process	A	B	C
p_0	7	4	3
p_1	❶	2	❶
p_2	6	0	0
p_3	0	1	1
p_4	4	3	1

Need

منابع آزاد $(2,3,0) \xrightarrow{p_1} (5,3,2) \xrightarrow{p_3} (7,4,3) \xrightarrow{p_4} (7,4,5) \xrightarrow{p_0} (7,5,5) \xrightarrow{p_2} (10,5,7)$

همانطور که مشاهده می شود ترتیب اجراء $< p_1, p_3, p_4, p_0, p_2 >$ یک ترتیب امن است . لذا اگر به درخواست های p_1 در لحظه t_0 پاسخ بگوئیم مطمئن هستیم که سیستم در حالت امن باقی خواهد ماند.

ب. اگر درخواست فوق را برآورده سازیم ماتریس های *Allocation* و *Need* به صورت زیر در خواهند آمد

process	A	B	C
p_0	7	4	3
p_1	1	2	2
p_2	6	0	0
p_3	0	1	1
p_4	❶	❶	1

Need

process	A	B	C
p_0	0	1	0
p_1	2	0	0
p_2	3	0	2
p_3	2	1	1
p_4	❸	❸	2

Allocation

(0,0,2)منابع آزاد

حال اگر منابع آزاد(بردار *Available*) را با هر سطر ماتریس *Need* مقایسه کنیم، هیچ کدام از سطرهای ماتریس *Need* از بردار مذکور کمتر نیست لذا این وضعیت نا امن می باشد