

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.

● **نقض گرفتن و منتظر ماندن:** برای نقض این شرط، باید مانع از ایجاد موقعیتی شد که در آن یک پردازش، 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 پردازش ها در ابتدای شروع اجرای پردازش 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 پردازش 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 پردازش بوده، پردازش دیگری می توانست از این استفاده کند. پس عیب این پروتکل کاهش بهره وری سیستم است.

B. هر پردازش 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 پردازش 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 پردازش دیگری استفاده شده است.

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

A. یکی این است که اگر پردازش ای درخواست منابعی را دارد که آن منابع آزاد نباشند، تمام منابع در اختیار پردازش ی درخواست کننده پس گرفته شوند. B. راه حل دیگر آن است که بررسی شود که آیا پردازش های منتظر تمام منابع مورد نیاز پردازش درخواست کننده را دارند یا نه، که اگر داشته باشند منابع از پردازش های منتظر گرفته شده و به پردازش درخواست کننده انتساب یابد و اگر پردازش یا پردازش های منتظر، منابع مورد نیاز پردازش ی درخواست کننده را نداشته باشند، خود پردازش ی درخواست کننده باید منتظر بماند و ممکن است در حین انتظار منابع در اختیارش نیز از وی گرفته شوند و به پردازش های دیگری انتساب یابند.

● **نقض انتظار پرفشی:**

جهت نقض انتظار پرفشی می بایست منابع را شماره گذاری کرد به طوری که هر پردازش بتواند منابع را در جهت صعودی شماره هایشان درخواست کند. به عنوان مثال اگر پردازش ای منبع شماره 3 را در اختیار داشته باشد، منبع شماره 1 را نمی تواند درخواست کند، ولی می تواند منبع شماره 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 از شماره ی چاپگر باشد. جهت اثبات اینکه روش گفته شده انتظار پرفشی را نقض می کند از برهان خلف استفاده می کنیم. فرض کنیم با اعمال این

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

$$f(R_0) < f(R_1) < \dots < f(R_n) < f(R_0)$$

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:

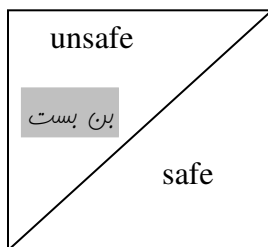
در این روش با توجه به اطلاعاتی نظیر حداقل نیاز پردازش ها به منابع و منابع تخصیص یافته به پردازش ها و موجودی، وقتی پردازش ای درخواست یک سری منابع را داشته باشد، اگر منابع موجود باشند، بررسی می کنیم که آیا با اجابت این درخواست سیستم به حالت امن می رود یا نه. اگر سیستم به حالت امن برود درخواست تخصیص داده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

حالت امن حالتی است که پردازش ها می توانند به ترتیب خاص منابع مورد نیازشان را گرفته و اجرایشان را با موفقیت پشت سر بگذارند.

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

تعریف دیگر: اگر سیستم بتواند منابع مورد درخواست را به ترتیبی تخصیص دهد که از بروز بن بست اجتناب شود کوئیم آن سیستم در حالت امن است.

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



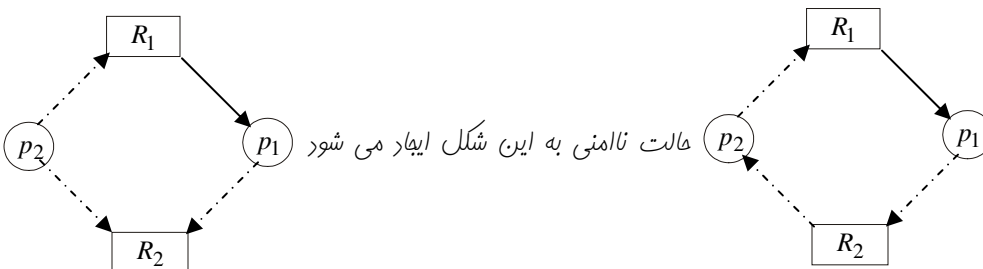
## برای اجتناب از بن بست دو راه وجود دارد:

1- از هر منبع فقط یک نمونه وجود داشته باشد: از گراف 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 منابع:** این گراف شبیه گراف تخصیص منابع است که علاوه بر یال های انتظار و تخصیص شامل یک یال ادعا (claim edge) می باشد که با نقطه چین نمایش داده می شود. به طوریکه  $(P_i) \cdots \rightarrow R_i$  بدین معناست که ممکن است در آینده پردازش  $P_i$  درخواست منبع  $R_i$  را داشته باشد. هر یال 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 طوری که ممکن است در آینده  $P_2$  طلب  $R_1$  و  $P_1$  طلب  $R_2$  را داشته باشد، که بن بست رخ می دهد



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

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

**الگوریتم بانکداران:** این الگوریتم اولین بار توسط دیسترا 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 تفویض نمی دهد و طوری عمل می کند که بتواند کلیه نیازهای مشتریانش را برآورده کند.

**ساقتمان داده های مورد نیاز (n تعداد پردازش ها و m تعداد منابع)**

### 1. $Max_{n \times m}$

$Max[i][j] = k$  بدین معناست که پردازش  $p_i$ ، جهت اجرای  $k$  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 پردازش  $p_i$  می باشد.

### 3. $Need_{n \times m}$

$Need[i][j] = k$  به این معناست که پردازش  $p_i$  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، پردازش  $p_i$  باشد الگوریتم به صورت زیر نوشته می شود.

1- اگر  $Need_i < Request_i$  (یعنی سطر مربوط به پردازش  $p_i$  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  $x \leq y$  اگر و فقط اگر  $x[i] \leq y[i]$  باشد به ازاء  $i = 1, 2, \dots, n$ . مثلا  
 $(0, 4, 2, 3) \leq (2, 8, 3, 5)$

2- اگر  $Request_i > Available$  باشد بدین معناست که منابع کافی جهت تفویض به پردازش  $p_i$ ، وجود ندارد، و  $p_i$  می بایست منتظر بماند.

3- وانمود می کنیم که منابع مورد درخواست پردازش  $p_i$ ، به وی اعطاء می شود، بنابراین ساقتمان داده ها به صورت زیر تغییر می یابند.

$$Alocation_i += Request_i$$

$$Need_i = Request_i$$

$$Available_i = Request_i$$

4- فراخوانی الگوریتم امنیت: اگر حالت سیستم ناامن باشد ساقتمان داده های تغییر یافته در قسمت 3 به حالت اول برگردانده 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 بردارهای  $work$  به طول  $m$  (متناظر با منابع) و  $Finish$  به طول  $n$  (متناظر با پردازش ها)

1-  $work = Available$  و برای  $i = 1 \dots n$  :  $Finish[i] = False$

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

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

3- زمانی به مرحله سه می آییم که پردازش ای پیدا شود که اجراش تمام نشده و با استفاده از منابع موجود می تواند نیاز هایش را بر آورده کرده و

اجراش را به اتمام برساند بنابراین پس از اتمام اجراش منابع در اختیارش را آزاد می سازد.  $work += Allocation$ ,  $finish[i] = true$

4- اگر برای تمام  $i$  ها  $Finish[i] = true$  ( $i = 1 \dots n$ ) باشد، بدین معناست که تمام پردازش ها می توانند با منابع موجود اجراشان 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 پردازش

نمی تواند اجراش 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 پردازش  $p_1$  الی  $p_5$  را داشته باشیم، با توجه به ماتریس های  $max$  و  $Allocation$  تشخیص دهید سیستم در حالت امن است یا

نه؟

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

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

Allocation

حل. ابتدا ماتریس  $Need$  را می سازیم با توجه به  $Need = Max - Allocation$  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  $Available$  را با توجه به فرمول  $sum -$  موجودی اولیه  $Available =$  ایبار می کنیم

و بر دار  $finish$  و  $work$  نیز در ابتدا به شکل زیر می باشند.

	$R_1$	$R_2$	$R_3$	$R_4$
work	1	2	2	2

	$p_1$	$p_2$	$p_3$	$p_4$	$p_5$
finish	f	f	f	f	f

	$R_1$	$R_2$	$R_3$	$R_4$
Available	1	2	2	2

پردازه هائی را که *false* هستند با *work* مقایسه می کنیم. (مقایسه سطرهای 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 برگرداند، کار را به همین ترتیب ادامه می دهیم.

	$R_1$	$R_2$	$R_3$	$R_4$		$P_1$	$P_2$	$P_3$	$P_4$	$P_5$
work	5	2	2	5	finish	f	f	T	f	f
work	7	2	3	6	finish	T	f	T	f	f
work	7	3	5	7	finish	T	T	T	f	f
work	7	5	6	7	finish	T	T	T	T	f
work	8	5	9	7	finish	T	T	T	T	T

ملاحظه می شود که سیستم در حالت امن است و ترتیب اجرا به صورت  $p_3, p_1, p_2, p_4, p_5$  می باشد در همین سؤال 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  $A, B, C$  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

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

max

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 فوراً پاسخ داده شود یا خیر.

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

حل.

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

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

Need

ب. ابتدا بردار Available را می سازیم

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

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

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

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
$p_1$	3	0	2
$p_2$	3	0	2
$p_3$	2	1	1
$p_4$	0	0	2

Allocation

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

Need

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

همانطور که مشاهده می شود ترتیب اجراء  $\langle p_1, p_3, p_4, p_0, p_2 \rangle$  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
$p_0$	0	1	0
$p_1$	2	0	0
$p_2$	3	0	2
$p_3$	2	1	1
$p_4$	3	3	2

Allocation

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

Need

(0,0,2) منابع آزاد

حال اگر منابع آزاد (بردار Available) را با هر سطر

ماتریس Need مقایسه کنیم، هیچ کدام از سطرها

ماتریس Need از بردار مذکور کمتر نیست لذا این

وضعیت نا امن می باشد

Pnu-Soal.ir