یادداشت‌های درس سیستم‌های عامل - بخش هشتم

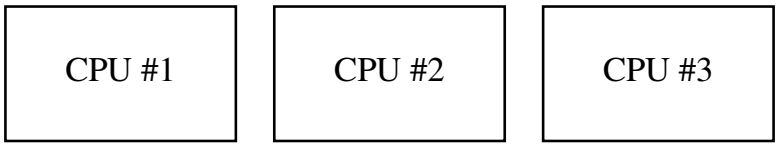
در بخش‌های قبل با بن‌بست آشنا شدیم. در این بخش بن‌بست را دقیق‌تر مطالعه می‌کنیم و روش‌های مقابله با آن را بررسی می‌نماییم.

- در مورد منابع در ادامه فرض می‌کنیم:

  الف) تعداد منابع محدود است.

  ب) هر منبع نوعی دارد (مثل پردازنده، چاپگر یا کارت صدا).

  ج) از هر نوع منبع ممکن است چند نمونه‌ی کاملا یکسان موجود باشند.



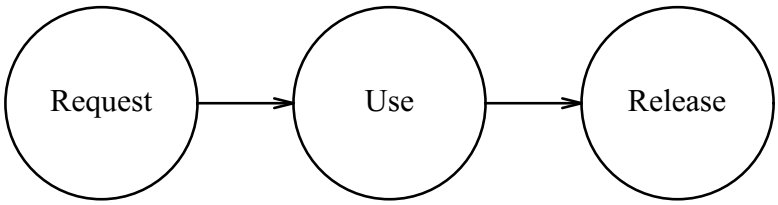
- برای مثال، فرض کنید در سیستم عاملی دو پردازنده موجود باشند. اگر برای پردازه‌ها این دو پردازنده تفاوتی نداشته باشند و با هر یک از این دو منبع پردازه‌ها بتوانند درخواستشان را خاتمه دهند این دو پردازنده دو نمونه از یک منبع هستند.
- دو چاپگر را که در مکان فیزیکی متفاوتی هستند (یا ویژگی‌های متفاوتی دارند)، نمی‌توان به عنوان دو نمونه از یک نوع منبع در نظر گرفت اگر برای استفاده کننده‌ها مهم باشد کدامیک به آنها تخصیص می‌یابد.
- قفل‌ها نیز منبع هستند؛ پردازه‌هایی که وارد ناحیه‌ی بحرانی می‌شوند قفل را در اختیار می‌گیرند و پس از اجرای ناحیه، قفل را آزاد می‌کنند.

- فرض می‌کنیم هر پردازه (یا ریسه) برای استفاده از هر منبع سه گام را انجام می‌دهد:

  الف) منبع را درخواست می‌کند.

  ب) از منبع استفاده می‌کند.

  ج) منبع را آزاد می‌کند.



- برای نمونه برای فایل‌ها:

  الف) فایل با فراخوانی open() درخواست می‌شود.

  ب) با توابعی مثل read() و write() به فایل دسترسی انجام می‌شود.

  ج) با تابع close() فایل بسته می‌شود.

- برای نمونه برای قفل‌ها:

  الف) با acquire() قفل درخواست می‌شود.

  ب) ناحیه‌ی بحرانی اجرا می‌شود.

  ج) با release() قفل آزاد می‌شود.

- برای نمونه برای حافظه:

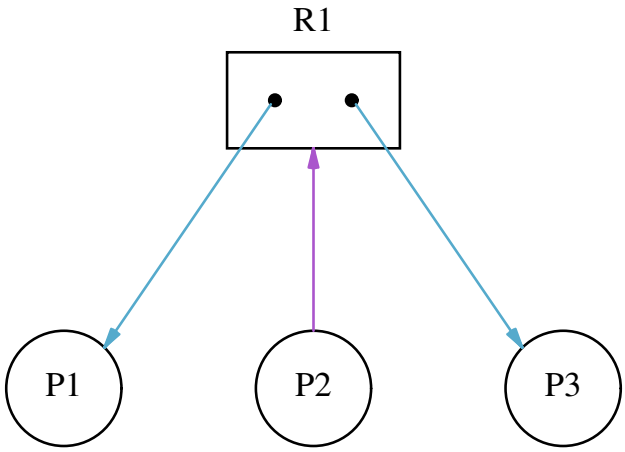
  الف) با فراخوانی توابعی مثل malloc() حافظه تخصیص می‌یابد.

  ب) از حافظه استفاده می‌شود.

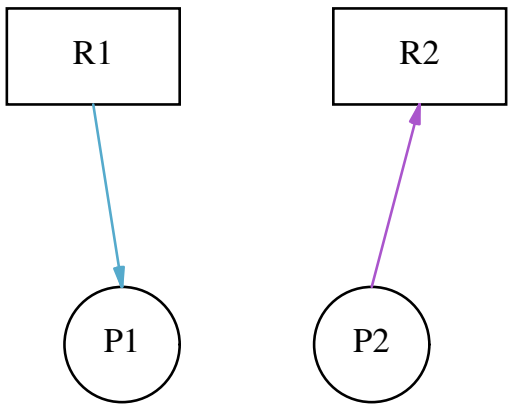
  ج) با توابعی مثل free() حافظه آزاد می‌شود.

گراف تخصیص منابع

- برای نمایش بهتر وضعیت سیستم، از گراف تخصیص منابع (Resource Allocation Graph) استفاده می‌کنیم.



- در این گراف به هر پردازه یک رأس به شکل دایره تخصیص می‌یابد.
- همچنین به هر نوع منبع یک مستطیل تخصیص می‌یابد؛ به تعداد نمونه‌های آن منبع، نقطه در داخل این مستطیل قرار می‌گیرد.
- یک یال از پردازه‌ی P به منبع R نشان می‌دهد که پردازه‌ی P منبع R را درخواست داده است.
- یک یال از یکی از نمونه‌های منبع R به پردازه‌ی P نشان می‌دهد که یک نمونه از منبع R به پردازه‌ی P تخصیص یافته است.

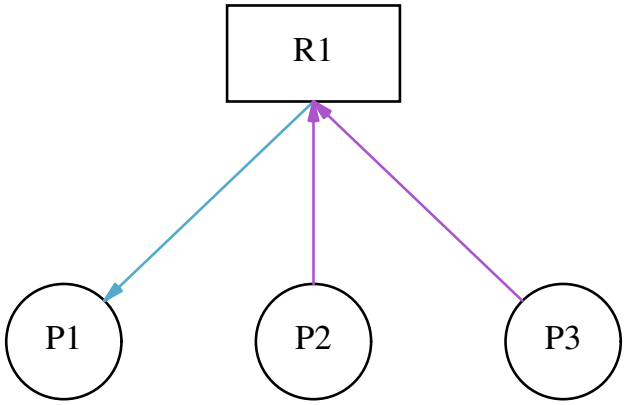


- اگر از منبعی فقط یک نمونه موجود باشد، می‌توان به جای قرار دادن نقطه در داخل آن، از خود مستطیل به جای نمونه استفاده نمود.

- چهار شرط لازم است تا بن‌بست رخ دهد.
- اگر هر یک از این شرایط موجود نباشد بن‌بست رخ نمی‌دهد.

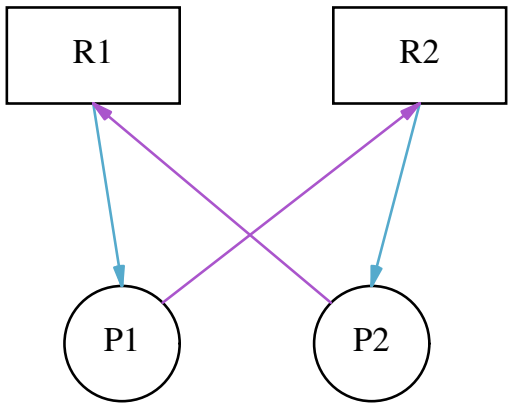
الف) انحصار متقابل

الف) انحصار متقابل (Mutual Exclusion): حداقل یکی از منابع باید به صورت انحصار متقابل استفاده شود (نباید قابل استفاده همزمان باشد).



ب) داشتن و انتظار

ب) داشتن و انتظار (Hold and Wait): برخی از پردازه‌ها باید در هنگام درخواست منبع، منبعی را در اختیار داشته باشند.



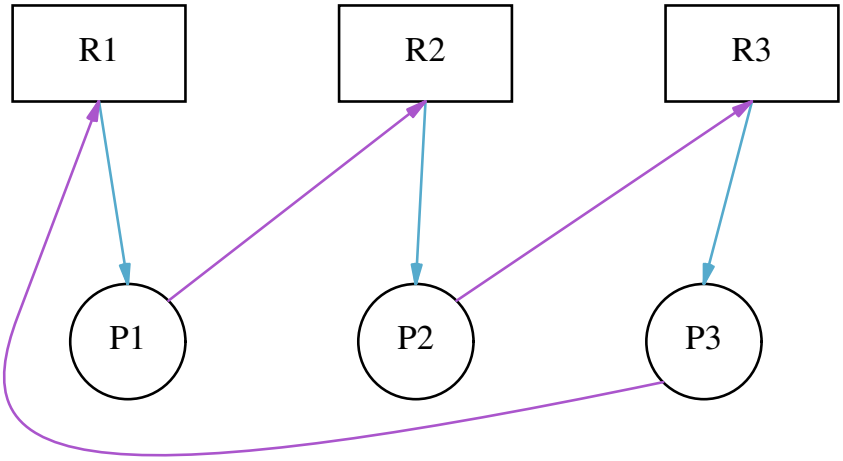
ج) نبودن Preemption

ج) نبودن Preemption (No Preemption): وقتی منبع در اختیار پردازه‌ای هست، آن منبع را نمی‌توان به زور از پردازه گرفت تا وقتی که خود پردازه آن را آزاد کند.

- برای نمونه وقتی پردازه‌ای قفلی را در اختیار دارد و در ناحیه‌ی بحرانی خود باشد، نمی‌توان قفل را از آن پردازه گرفت و به پردازه‌ی دیگری داد؛ اگر این کار انجام شود وضعیت رقابتی رخ می‌دهد. با منابعی به این صورت بن‌بست رخ می‌دهد.
- اگر پردازنده به پردازه‌ای اختصاص داده شده باشد، می‌توان آن را از پردازه گرفت، آن را به پردازه‌های دیگر داد و بعدا به پردازه برگرداند. اگر همه‌ی منابع به این صورت باشند، بن‌بست رخ نمی‌دهد.

د) انتظار چرخشی

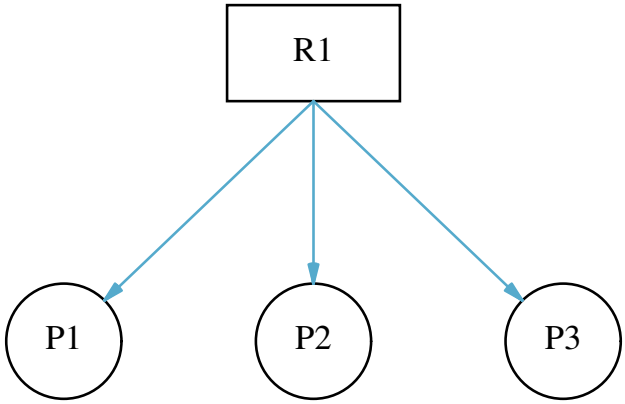
د) انتظار چرخشی (Circular Wait): باید دنباله‌ای از پردازه‌ها موجود باشند که هر پردازه منتظر پردازه‌ی بعدی خودش هست و پردازه‌ی آخر نیز منتظر پردازه‌ی اول.



- روش‌های متفاوتی برای مقابله با بن‌بست وجود دارد؛ در ادامه آنها را بررسی می‌کنیم.
- الف) تغییراتی ایجاد کنیم که هیچ‌گاه بن‌بست رخ ندهد.
- ب) اجازه دهیم بن‌بست رخ دهد، سپس آن را تشخیص می‌دهیم و سپس بازیابی می‌کنیم.
- ج) مشکل بن‌بست را نادیده می‌گیریم.

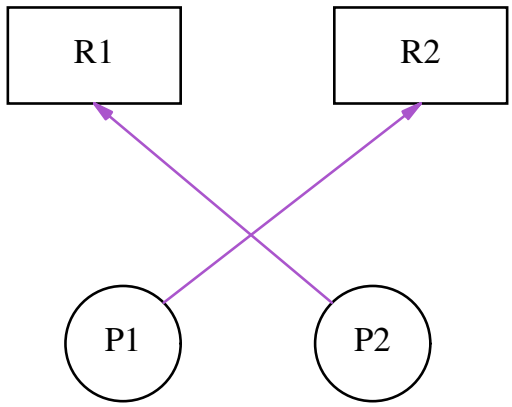
- در پیشگیری از بن‌بست (Deadlock Prevention) یکی از شرایط لازم برای بن‌بست را نقض می‌کنیم تا هیچ‌گاه بن‌بست رخ ندهد.
- هر یک از چهار شرط لازم برای بن‌بست را بررسی می‌کنیم که در چه شرایطی قابل نقض شدن هستند.

- انحصار متقابل وقتی نقض می‌شود که منبع بتواند به صورت همزمان در اختیار چند پردازه قرار بگیرد.



- قطعا برای برخی از انواع منابع این کار ممکن نیست: برای مثال نمی‌توان به صورت همزمان چاپگر یا CD Writer را در اختیار دو پردازه قرار داد. یا تخصیص دادن یک قفل به چند پردازه ممکن است وضعیت رقابتی ایجاد کند.
- اما منابعی هم هستند که استفاده‌ی همزمان از آنها مشکلی ایجاد نمی‌کند، مثل فایل‌های فقط خواندنی (Read-only).

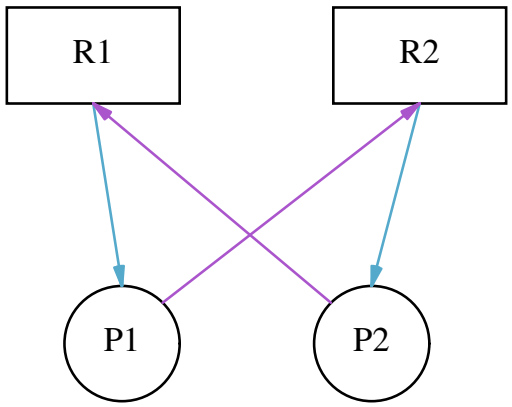
- برای اینکه این شرط نقض شود لازم است پردازه‌ها در زمان درخواست یک منبع، هیچ منبعی را در اختیار نداشته باشند.



- راه اول: پردازه‌ها در هنگام درخواست منبع، همه‌ی منابعی که در اختیار دارند را آزاد کنند و منابعی که آزاد کرده‌اند و منابع جدید را با هم درخواست دهند.
- راه دوم: پردازه‌ها همه‌ی منابعی که در زمان اجرا ممکن است نیاز داشته باشند در هنگام شروع از سیستم عامل درخواست کنند و در زمان اجرا منبع دیگری را درخواست ندهند.
- بدی این دو روش امکان بروز گرسنگی است: ممکن است برای از منابع مورد نیاز پردازه بارها تخصیص داده شوند و آزاد شوند ولی به خاطر اینکه همه‌ی منابع به صورت همزمان آزاد نیستند، پردازه نمی‌تواند اجرا شود.
- یک بدی روش دوم این است که منابع ممکن است مدت زیادی بدون استفاده بمانند. برای مثال، اگر پردازه‌ای فقط در زمان کوتاهی از اجرایش به گرفتن یک قفل احتیاج داشته باشد، مجبور است قفل را از شروع تا پایان اجرایش در اختیار داشته باشد (سایر پردازه‌ها برای استفاده از آن باید منتظر شوند).

نقض No Preemption

- برای نقض شرط No Preemption باید بتوان هر منبعی را از پردازه‌های منتظر گرفت و به سایر پردازه‌ها داد.
- شکل زیر را در نظر بگیرید.

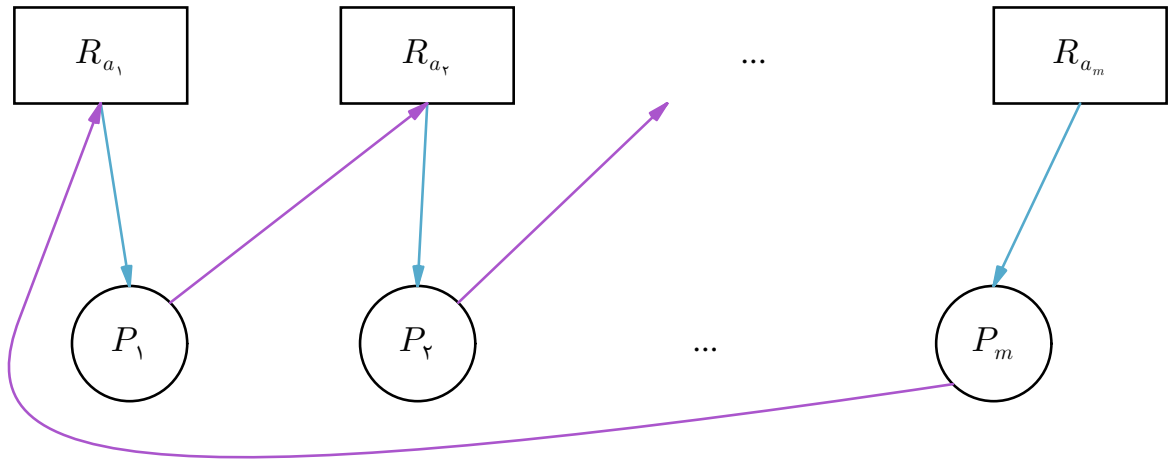


- در این حالت بن‌بست رخ داده است.
- اگر Preemption برای منبع R1 ممکن باشد: چون پردازه‌ی P1 منتظر است، سیستم عامل منبع R1 را از P1 می‌گیرد و به P2 می‌دهد.
- چون P2 هر دو منبع مورد نیازش را در اختیار می‌گیرد، پردازشش را ادامه می‌دهد و در پایان، هم R1 و هم R2 را آزاد می‌کند.
- سیستم عامل R1 را به P1 بر می‌گرداند و R2 را نیز به آن تخصیص می‌دهد.
- در نتیجه هر دو پردازه بدون بروز بن‌بست خاتمه می‌یابند.
- اما Preemption فقط برای برخی از منابع امکان دارد: منابعی که بتوان وضعیت آن را (در هنگام گرفتن آن از یک پردازه) ذخیره کرد و بعدا آن وضعیت را (در هنگام بازگرداندن به پردازه) برگرداند.
- برای برخی از منابع مثل پردازنده و حافظه این کار ممکن است ولی برای بسیاری از منابع مثل قفل‌ها، چاپگر و کارت صدا این کار معمولا امکان ندارد.

- برای نقض انتظار چرخشی می‌توانیم از روش زیر استفاده کنیم.

- به هر منبع موجود یک عدد تخصیص دهید.
- پردازه‌ای می‌تواند منبعی را درخواست دهد که شماره‌ی آن منبع از شماره‌ی همه‌ی منابعی که در اختیار آن پردازه هست بزرگ‌تر باشد.
- برای مثال، اگر پردازه‌ای منبع R2 را در اختیار داشته باشد، اجازه ندارد منبع R1 را درخواست دهد (چون شماره‌ی R1 کوچک‌تر از شماره‌ی R2 است) اما می‌تواند منبع R5 را درخواست دهد (چون شماره‌ی R5 بزرگ‌تر از شماره‌ی R2 هست).
- اگر این پردازه بخواهد R1 را تخصیص دهد، باید اول R2 را آزاد کند، سپس R1 را تخصیص دهد و پس از آن R2 را در اختیار بگیرید.

- برای اینکه نشان دهیم با این روش انتظار چرخشی رخ نمی‌دهد، از برهان خلف استفاده می‌کنیم.
- فرض کنید چرخه‌ی انتظاری مشابه چرخه‌ی زیر با طول m ایجاد شود.



- چون پردازه‌ی P_1 در هنگام درخواست R_{a_2} منبع R_{a_1} را در اختیار داشته است، داریم $a_2 \prec a_1$.
- به صورت مشابه چون پردازه‌ی P_2 در هنگام درخواست R_{a_3} منبع R_{a_2} را در اختیار داشته است، داریم $a_3 \prec a_2$.
- این چرخه ادامه دارد تا به آخرین پردازه می‌رسیم. به صورت مشابه چون پردازه‌ی P_m در هنگام درخواست R_{a_1} منبع R_{a_m} را در اختیار داشته است، داریم $a_1 \prec a_m$.
- در نتیجه رابطه‌ی زیر را خواهیم داشت که یک تناقض است.

$$a_1 \prec a_2 \prec \cdots \prec a_m \prec a_1$$

- پس فرض خلف باطل است و انتظار چرخشی نمی‌تواند رخ دهد.

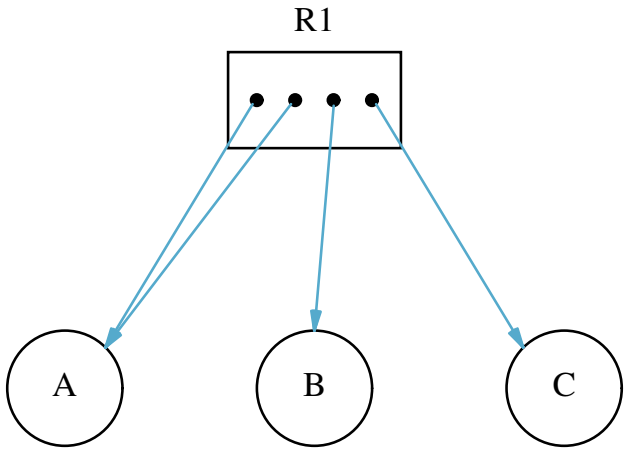
- روشی که برای نقض انتظار چرخشی بررسی کردیم محدودیت‌هایی دارد که باعث می‌شود این روش فقط گاهی قابل استفاده باشد.
- گاهی تعیین ترتیب برای گرفتن منابع ممکن نیست و منابع در ترتیب‌های متفاوتی توسط پردازه‌ها درخواست می‌شوند.
- گاهی شماره‌ی منابعی که باید درخواست شود در طول زمان و به صورت پویا مشخص می‌شود. برای مثال شماره‌ی حساب‌هایی که در یک تراکنش شرکت می‌کنند در زمان اجرا مشخص می‌شود.

- در اجتناب از بن‌بست (Deadlock Avoidance)، سیستم عامل به اطلاعات بیشتری در مورد منابع و پردازه‌ها احتیاج دارد.
- این اطلاعات بیشتر معمولا حداکثر تعداد نمونه‌هایی است که هر پردازه می‌تواند از هر نوع منبع درخواست دهد.
- با کمک این اطلاعات و وضعیت فعلی سیستم، سیستم عامل سعی می‌کند همواره سیستم را در وضعیت امن نگه داشته باشد.
- مفهوم وضعیت امن در ادامه مشخص می‌شود.

- جدول زیر اطلاعات سه پردازه را نشان می‌دهد.

Process	R1 (current/maximum)
A	2/3
B	1/3
C	1/1

- در ستون دوم دو عدد نوشته شده است: عدد اول تعداد منابعی از نوع R1 است که به هر پردازه تخصیص داده شده است. عدد دوم حداکثر تعداد منابعی از نوع R1 است که هر پردازه می‌تواند تخصیص دهد.
- فرض تعداد کل نمونه‌های منبع R1 چهار باشد.



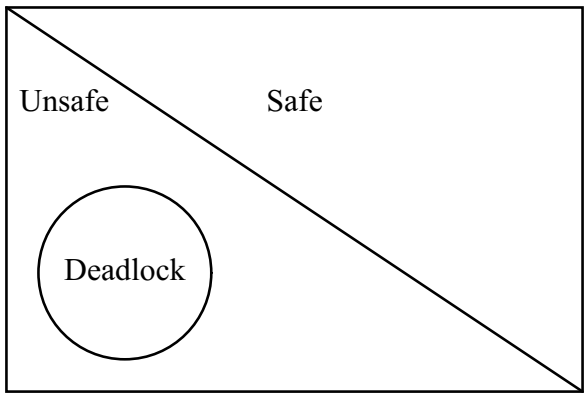
- یک دنباله از پردازه‌ها امن است اگر مطابق با ترتیب این دنباله هر پردازه بتواند حداکثر منابع مورد نیازش را درخواست دهد و کارش را تمام کند.
- برای نمونه در مثال بالا دنباله‌ی <C, A, B> را در نظر بگیرید (عنصر اول C است)؛ این دنباله امن است.
- اگر برای وضعیتی از سیستم یک دنباله‌ی امن وجود داشته باشد، وضعیت سیستم امن است ولی اگر هیچ دنباله‌ی امنی موجود نباشد وضعیت سیستم ناامن است.

- برای مثال وضعیت پردازه‌های زیر را در نظر بگیرید.

Process	R1 (current/maximum)
A	2/4
B	1/3
C	1/1

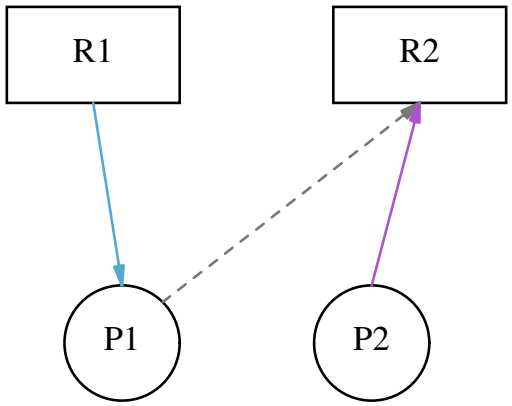
- اگر فقط چهار نمونه از منبع R1 موجود باشند، هیچ دنباله‌ی امنی برای این پردازه‌ها موجود نیست و در نتیجه وضعیت سیستم ناامن است.

- دقت کنید که وضعیت ناامن به مفهوم رخداد بن‌بست نیست.
- اگر وضعیت سیستم در حالت ناامن قرار داشته باشد، احتمال دارد بن‌بست رخ دهد.



- در اجتناب از بن‌بست وضعیت سیستم را همواره در حالت امن قرار می‌دهیم تا بن‌بست رخ ندهد.

- با گراف تخصیص منابع آشنا شدیم؛ برای استفاده از آن در اجتناب از بن‌بست، لازم است یال جدیدی به آن اضافه شود.
- یال Claim از پردازه P به منبع R نشان می‌دهد که ممکن است پردازه‌ی P منبع R را در آینده درخواست دهد.
- یال Claim را با خط چین نمایش می‌دهند.
- برای نمونه، در شکل زیر یال Claim از پردازه‌ی P1 به منبع R2 نشان می‌دهد که در آینده ممکن است پردازه‌ی P1 منبع R2 را درخواست دهد.



- گراف تخصیص منابع برای اجتناب از بن‌بست تنها وقتی مناسب است که از هر منبع یک نمونه داشته باشیم.
- سیستم عامل با استفاده از الگوریتم زیر به هر درخواست پاسخ می‌دهد.

الگوریتم سیستم عامل برای پاسخ به درخواست در اجتناب از بن‌بست با کمک گراف تخصیص منابع

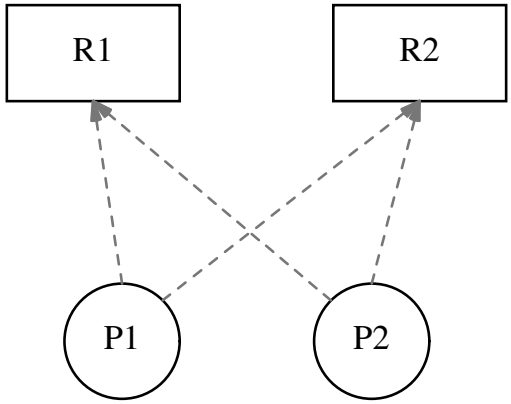
ورودی: پردازه‌ی P منبع R را درخواست می‌کند.

خروجی: تخصیص منبع یا متوقف کردن پردازه.

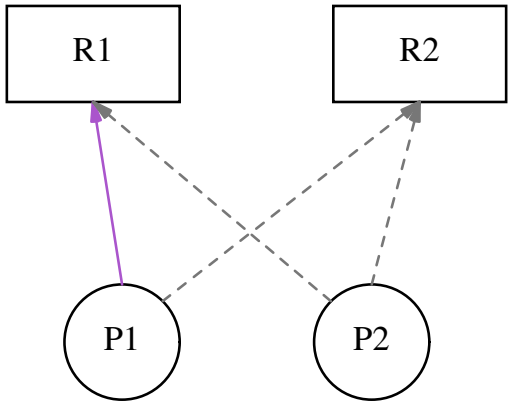
۱ یال Claim مربوط به این درخواست را به یال درخواست از P به R تبدیل کن.
۲ اگر R آزاد بود
۳ یال درخواست را با یال تخصیص از R به P جایگزین کن.
۴ اگر گراف دور نداشت
۵ وضعیت امن است.
۶ تخصیص را انجام بده.
۷ اگر گراف دور داشت
۸ وضعیت ناامن است.
۹ یال تخصیص را با درخواست از P به R جایگزین کن.
۱۰ تخصیص را انجام نده و پردازه را در حالت انتظار قرار بده.
۱۱ در غیر این صورت
۱۲ پردازه را در حالت انتظار قرار بده.

گراف تخصیص منابع - مثال

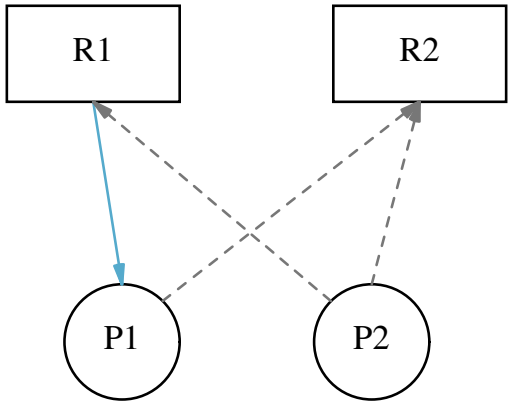
- دو پردازه و دو منبع را در نظر بگیرید.



- فرض کنید پردازه‌ی P1 منبع R1 را درخواست کند.

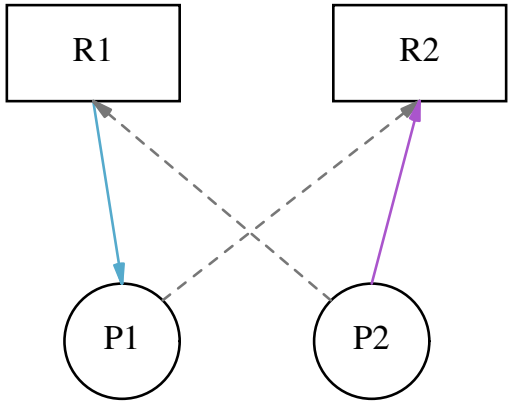


- اگر تخصیص انجام شود گراف به صورت زیر تغییر می‌کند.

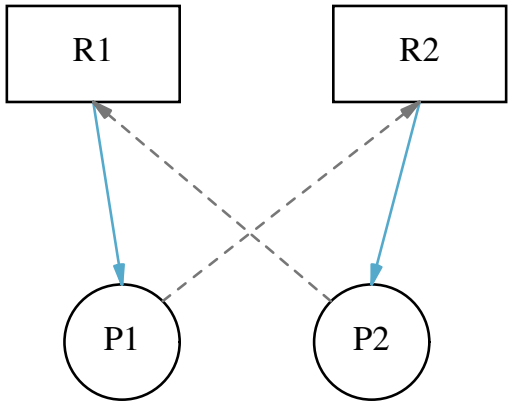


- چون این گراف دو ندارد، وضعیت امن است و تخصیص انجام می‌شود.

- فرض کنید پردازه‌ی P2 منبع R2 را درخواست کند.



- اگر تخصیص انجام شود گراف به صورت زیر تغییر می‌کند.



- چون این گراف دور دارد، وضعیت ناامن است و تخصیص نمی‌تواند انجام شود؛ یال به درخواست تبدیل می‌شود و پردازه منتظر می‌شود.