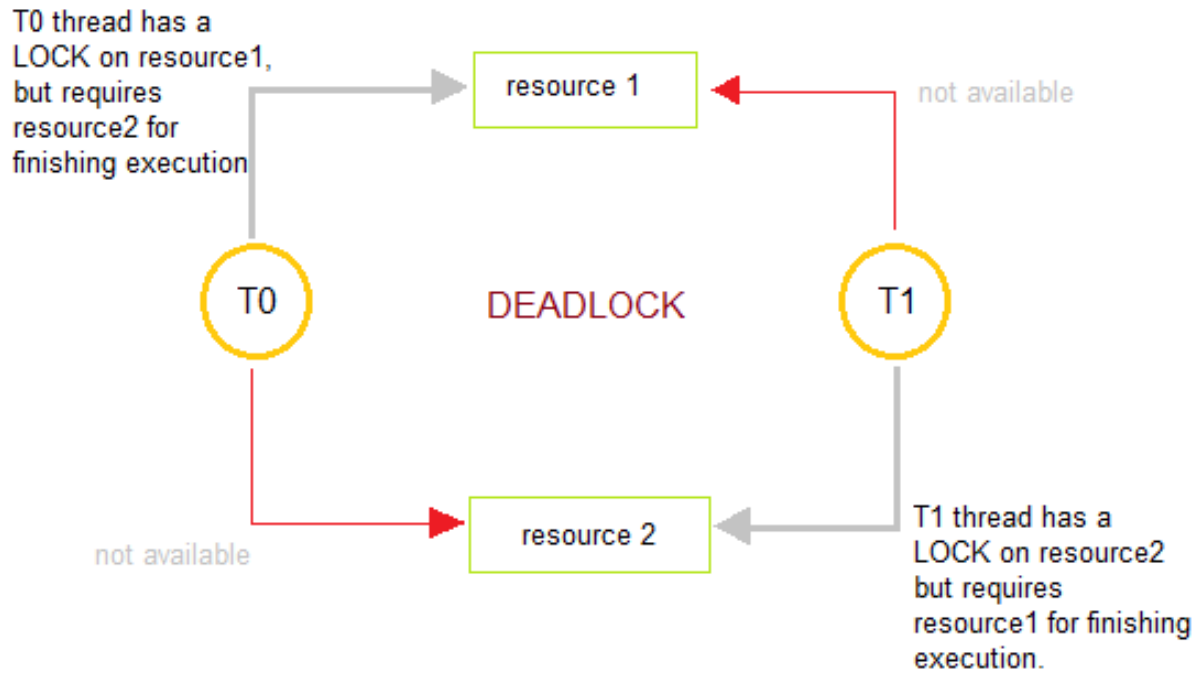
سیستم عامل جلسه 8

بن بست‌ها (Dead locks)

در محیط چند برنامه ای ممکن است چندین فرایند برای تعداد محدودی از منابع با هم رقابت کنند. فرایند، منابعی را درخواست می‌کند و چنان چه این این منابع در آن زمان فراهم نباشد، فرایند به حالت انتظار می‌رود. ممکن است منابع درخواستی این فرایند دراختیار فرایند دیگری باشند که در حال انتظار هستند و این فرایند هرگز از حالت انتظار خارج نشود این وضعیت را بن بست می‌گویند

In operating systems, a deadlock is a situation where two or more processes are unable to continue executing because they are waiting for each other to release resources. Essentially, it's a state where each process is stuck and cannot proceed because it needs access to a resource that is being held by another process, which in turn is waiting for a resource that the first process is holding.



شاید بهترین مفهوم از بن بست، از قانونی استنباط شود که قانون گذاران به تصویب رساندند. بخشی از قانون می‌گوید "وقتی دو قطار در یک تقاطع به هم نزدیک می‌شوند و هر دو بایدکاملا باستند و هیچ کدام نباید حرکت کنند، مگر این که دیگری رفته باشد".

در این فصل، روش‌هایی را توصیف می‌کنیم که سیستم عامل می‌تواند برای جلوگیری از بن بست یا اداره کردن آن به کار گیرد. گرچه بعضی از برنامه‌های کاربری می‌توانند تشخیص دهند چه برنامه‌هایی دچار بن بست می‌شوند، ولی سیستم‌های عامل امکاناتی برای پیشگیری از بن بست ندارند و برنامه نویس باید تضمین کند که برنامه‌های آن‌ها فاقد بن بست باشد. با توجه به گرایش های فعلی؛ مثل تعداد زیاد فرایندها، برنامه‌های چند نخی (multi thread)، منابع زیاد در یک سیستم و تاکیدبر سرورهای پایگاه داده و فایل‌هایی با طول عمر زیاد به جای سیستم‌های دسته ای، مساله های بن بست در حال متداول شدن هستند.

مدل سیستم

هر سیستم متشکل از تعداد محدودی از منابع است که باید بین فرایندهای متقاضی و رقیب توزیع شود. این منابع به چندین نوع تقسیم می‌شوند که هر کدام ممکن است شامل چند نمونه‌ی یکسان باشند

فضای حافظه، چرخه‌های پردازنده، فایل‌ها، دستگاه‌های i/o (مثل چاپگر و گرداننده‌های DVD) از انواع منابع(Resource) هستند.

اگر فرایند نمونه ای از یک نوع منبع را درخواست کند، تخصیص هر نمونه از آن نوع، ان درخواست را برآورده می‌کند. اگر درخواست برآورده نشود، ان گاه نمونه‌ها یکسان نیستند و نوع منابع به طور مناسب دسته بندی نشده اند. به عنوان مثال، یک سیستم ممکن

است دو چاپگر داشته باشد و اگر برای کسی مهم نباشد که کدام چاپگر خروجی را تولید می‌کند،این دو چاپگر ممکن است در یک دسته از منبع قرار گیرند. اما اگر یک چاپگر در طبقه‌ی نهم و چاپگر دیگر در طبقه‌ی همکف باشد، افرادی که در طبقه‌ی نهم قرار دارند می‌دانند که در هر دو چاپگر یکسان عمل نمی‌کنند و در نتیجه لازم است برای هر چاپکر دسته‌ی جداگانه ای از منبع تعریف کرد .هر فرایند قبل از به کارگیری منبعی، باید آن رادرخواست کند و پس از استفاده از آن، باید آن را رها کند. هر فرایند برای انجام وظیفه اش ممکن است چندین منبع را درخواست کند بدیهی است که تعداد منابع درخواستی نباید بیش از منابع موجود در سیستم باشد.

در عملیات عادی هر فرایند ممکن است فقط به ترتیب زیر از یک منبع استفاده کند:

1.درخواست

اگر درخواست نتواند فوراً عملی شود (مثلاً منبع ر اختیار فرایند دیگری باشد)، فرایند درخواست کننده باید منتظر بماند تا منبع را در اختیار بگیرد

2.به کار گیری

فرایند می‌تواند از منبع استفاده کند (مثلاً اگر منبع درخواستی چاپگر باشد، می‌تواند عمل چاپ را انجام دهد)

3.آزاد کردن

فرایند منبع را آزاد می‌کند.

درخواست و آزاد سازی منابع، فراخوان‌های سیستم هستند. نمونه‌هایی از فراخوان‌های سیستم عبارتند از: ()request و ()release برای دستگاه‌ها، ()open و ()close برای فایل‌ها، ()allocate و ()free برای حافظه، درخواست و آزادی سازیِ منابعی که تحت مدیریت سیستم عامل نیستند. از طریق عملیات ()wait و ()signal بر روی سمافورها یا از طریق به درست آوردن و آزاد سازی قفل انحصار متقابل (mutex) صورت می‌گیرد. برای هر استفاده‌ی فرایند یا نخ از از منبع تحتِ مدیریتِ هسته، سیستم عامل بررسی می‌کند تا مطمین شود که فرایندی منبعی را درخواست کرده باشد و منبع به آن تخصیص داده شود، یک جدول در سیستم، ثبت می‌کند که کدام منبع آزاد و کدام فرایند تخصیص یافته است. اگر فرایندی منبعی را درخواست کند که فعلاً به فرایند دیگری تخصیص یافته باشد و می‌تواند به صف فرایندهای منتظر آن منبع اضافه شود.

مجموعه ای از فرایندها وقتی در حالت بن بست قرار دارند که هر فرایند موجود در آن مجموعه، منتظر رویدادی باشند که فقط به وسیله‌ی یک فرایند دیگر از ان مجموعه رخ خواهد داد.

منابع ممکن است فیزیکی باشند مثل چاپگرها، گرداننده‌های نوار، فضای حافظه، و چرخه‌های پردازنده، یا ممکن است ما فایل‌ها، سمافورها، ناظرها

برای تشریح حالت بن بست، سیستمی با سه گرداننده‌ی CD RW را در نظر بگیرید. سه فرایند هر کدام یکی از سه گرداننده‌ی CD RW را در اختیار دارد. اگر هر فرایند، گرداننده دیگری را درخواست کند، این سه فرآیند در حالت بن بست خواهد بود. هر کدام منتظر آزاد شد CD RW هستند که فقط هر کدام از این فرایندهای منتظر می‌توانند آزاد کنند

این مثال بن بستی را نشان می‌دهد که شامل یک نوع منبع است

بن بست ممکن است شامل انواع مختلفی از منابع باشد به عنوان مثال، سیستمی با یک چاپگر و یک گرداننده‌ی DVD را در نظر بگیرید . فرض کنید فرایند P1گرداننده‌ی DVD و فرایند P2 چاپگر را در اختیار دارد. اگر P1 چاپگر را درخواست کند و P2 گرداننده DVD را خواست کند، بن بست رخ می‌دهد

مشخصات بن بست

در بن بست، اجرای فرایندها خاتمه پیدا نمی‌کند، منابع سیستم به هم گره می‌خورند و از کارهای دیگر جلوگیری می‌شود. قبل از پرداختن به راه حل‌های اداره کردنِ مساله ی بن بست، در تعیین کننده‌ی بن بست‌ها را توصیف می‌کنیم.

شرایط ضروری Condition Necessary for deadlock

وضعیت بن بست در صورتی پیش می‌آید که چهار شرط زیر همزمان در یک سیستم وجود داشته باشد

1.انحصار متقابل Mutual Exclusion

حداقل یک منبع باید در حالت غیر اشتراکی نگهداری شود، یعنی در هر زمان فقط یک فرایند می‌تواند از آن منبع استفاده کند. اگر فرایند دیگری آن منبع را درخواست کند، فرایند درخواست کننده باید منتظر بماند تا آن منبع آزاد شود – مثال منبع چاپگر یا مانیتور در حالت تمام صفحه – مثال در فرایند تمام صفحه تنها یک فرایند می تواند حالت تمام صفحه را بگیرد و همزمان دو فرایند نمی توانند به صورت تمام صفحه اجرا شوند

2.نگهداری و انتظار Hold and Wait

باید فرایندی وجود داشته باشد که حداقل یک منبع را در اختیار داشته باشد و منتظر به دست آوردن منبع دیگری باشد که فعلاً در اختیار فرایند دیگری است

3.بدون قبضه کردن Non preemptive

منابع نمی‌توانند قبضه شوند، یعنی آزادسازی منبع به عهده فرایند است که آن را در اختیار دارد و پس از کامل کردن وظیفه‌ی خود، ان را آزاد می‌کند

4.انتظار چرخشی Circular wait

باید مجموعه ای از فرایندهای منتظر {p0,p1,p2,p3, ...} وجود داشته باشند که p0 منتظر منبعی باشد که در اختیار p1 است و p1 منتظر منبعی باشد که در اختیار p2 است و به همین ترتیب p_{n-1} منتظر منبعی است که در اختبار، p_n است و p_n منتظر منبعی است که در اختیار p0 است

برای وقوع بن بست هر چهار شرط باید وجود داشته باشد.

گراف تخصیص منابع (system resource allocation graph)

بن بست‌ها می‌توانند توسط گراف‌های جهت داری به نام گراف تخصیص منابع سیستم (system resource allocation graph) ، دقیق‌تر تشریح شوند، این گراف شامل مجموعه ای از راس ها به نام V و مجموعه ای از یال‌ها به نام E است

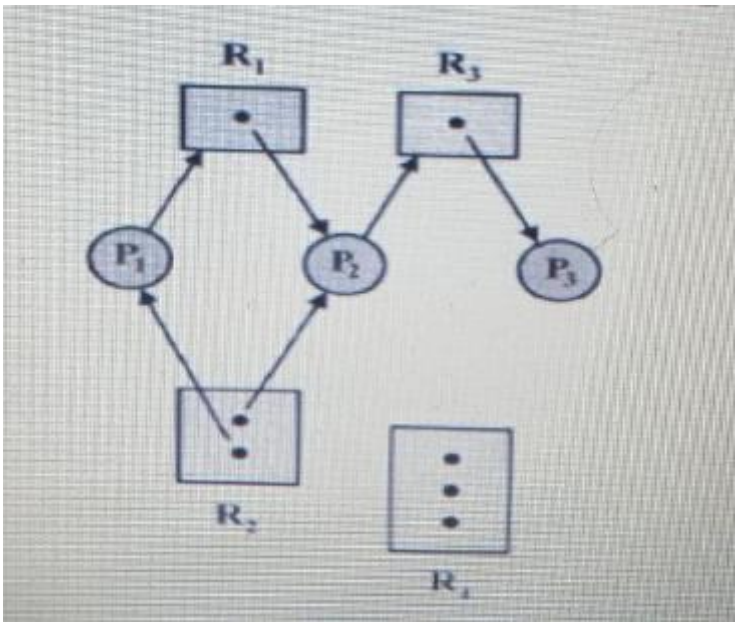
مجموعه‌ی V به دو نوع گره‌های مختلف تقسیم می‌شود:

مجموعه‌ی $P = \{P1,P1,P3, ..., Pn\}$ حاوی تمام فرآیند های فعال در سیستم است و مجموعه‌ی $R = \{R1, R2, R3, ..., Rm\}$ حاوی انواع منابع موجود در سیستم عامل است.

یک یال جهت دار از فرایند pi به منبع Rj به صورت $P_i \rightarrow R_j$ نشان داده می‌شود. این علامت نشان می‌دهد که فرایند pi نمونه ای منبع نوع Rj را درخواست کرده است و منتظر آن منبع است. یک یال جهت دار از منبع نوع Rj به فرایند Pi به صورت $R_j \rightarrow P_i$ نمایش داده می‌شود و مشخص می‌کند که نمون از منبع نوع Rj به فرایند pi تخصیص یافته است. یال جهت دار $P_i \rightarrow R_j$ یک یال درخواست (Request edge) نام دارد و یال جهت دارد $R_j \rightarrow P_i$ یال تخصیص (Assignment edge) نام دارد.

در نمایش تصویری، هر فرایند Pi را با یک دایره و هرنوع منبع Rj را با یک مربع نمایش می‌دهیم. چون منبع Rj ممکن است چند نمونه داشته باشد و هر یک از نمونه‌ها را به صورت نقطه ای در مربع نمایش می‌دهیم. توجه کنید که یال درخواست ممکن است چند نمونه داشته باشد، هر یک از نمونه‌ها را به صورت نقطه ای در مربع نمایش می‌دهیم، توجه کنید که یال درخواست فقط به مربع Rj اشاره می‌کند در حال که یال تخصیص باید به یکی از نقاط درون مربع نیز تخصیص یابد.

وقتی فرایند Pi نمونه ای از منبع نوع Rj درخواست می‌کند، یک یال درخواست به تخصیص منابع اضافه می‌شود، در صورتی که این درخواست انجام پذیر باشد، این یال درخواست فوراً به یک یال تخصیص تبدیل می‌شود، وقتی فرایندی به منبعی نیاز نداشته باشد، آن را آزاد می‌کند و در نتیجه یال تخصیص حذف می‌شود



گراف تخصیص منابع که شکل بالا آمده است وضعیت زیر را نشان می‌دهد

مجموعه‌های E و R و P که عبارتند از :

P = {P1, P2, P3}

R = {R1, R2, R3, R4}

E={P1 → R1, P2 → R3, R1 → P2, R2 → P2, R2 → P1, R3 → P3}

نمونه‌ی منابع:

- یک نمونه از منبع نوع R1
- دو نمونه از منبع R2
- یک نمونه از منبع R3
- سه نمونه از منبع نوع R4

حالت‌های فرایند:

- فرایند P1 نمونه از منبع نوع R1 را در اختیار دارد و منتظرنمونه ی از منبع نوعR1 است.
- فرایند P2 یک نمونه از R1وR2 را در اختیار دارد و منتظر نمونه ای از منبع نوعR3 است.
- فرایند P3 یک نمونه از منبع R3 را در اختیار دارد.

با توجه به تعریف گراف تخصیص منابع، می‌توان نشان داد که:

اگر گراف فاقد **چرخه (دور)** باشد، هیچ فرایندی در بن بست نیست

از طرف دیگر، اگر گراف حاوی چرخه باشد، ممکن است بن بست وجود داشته باشد.

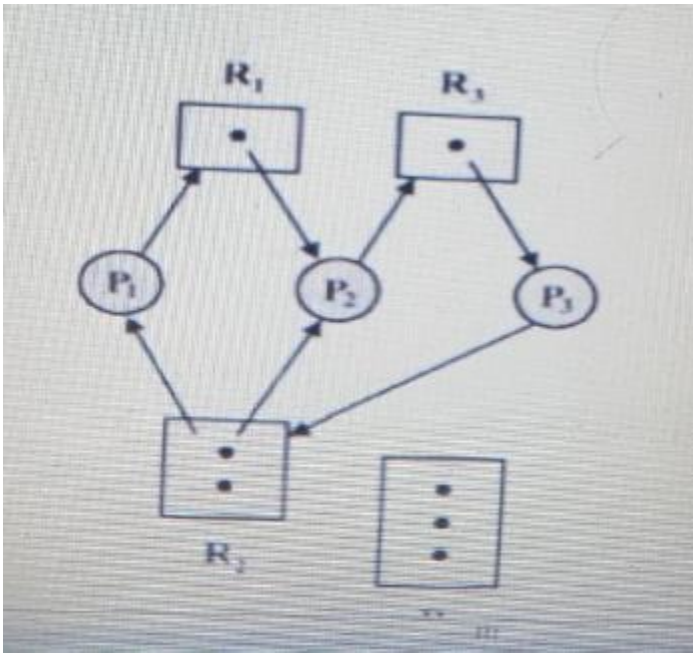
اگر هر نوع منبع دقیقاً یک نمونه داشته باشند. چرخه، نشان دهنده‌ی وجود بن بست است.

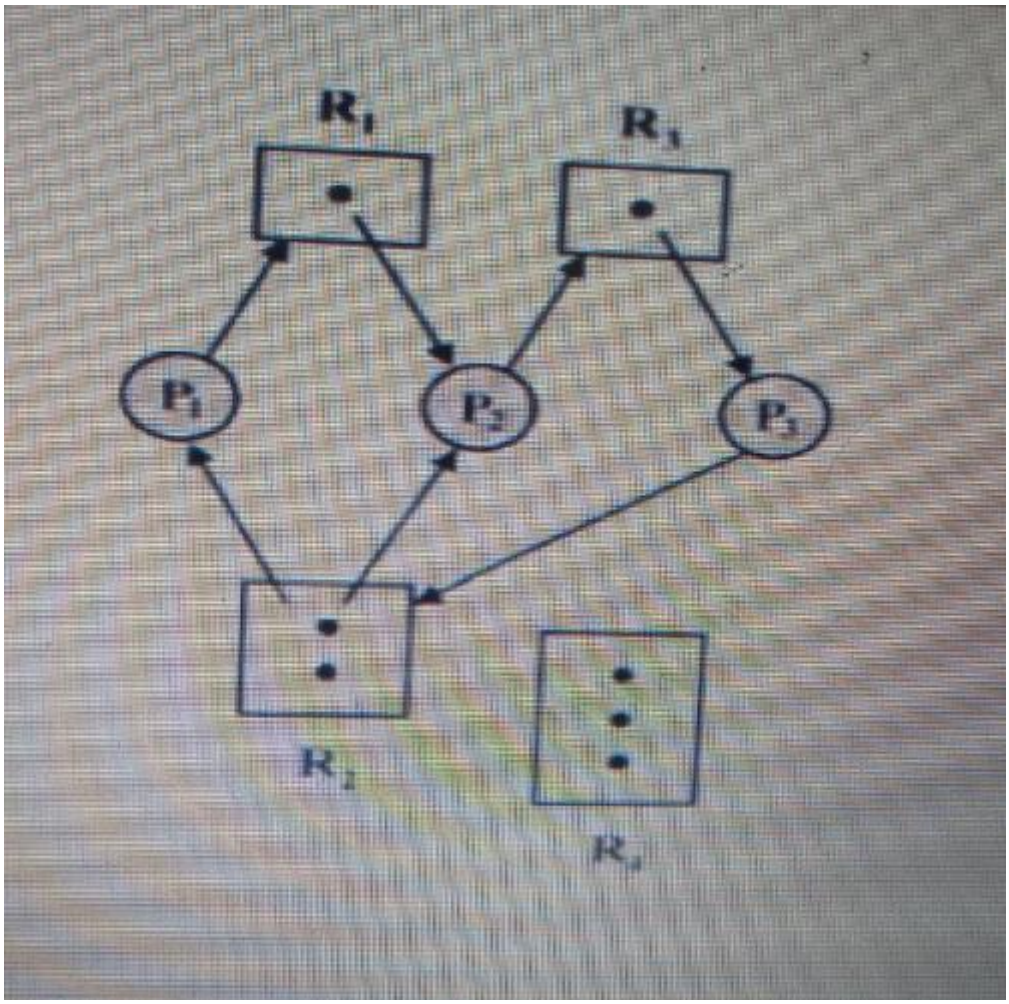
اگر چرخه، حاوی فقط یک مجموعه از انواع منبع باشد و هر نوع منبع شامل یک نمونه باشد، بن بست وجود دارد.

هر فرایند موجود در چرخه، در بن بست قرار دارد و در این حالت، چرخه ای در گراف، شرط لازم و کافی برای وجود بن بست است.

اگر هر نوع منبع شامل چند نوع نمونه باشد، وجود چرخه الزاماً به معنای وجود بن بست نیست، در این حالت، وجود چرخ‌های را شرط لازم برای وجود بن بست است ولی شرط کافی نیست.

برای تشریح این مفهوم، گراف تخصیص منابع شکل قبلی را در نظر می‌گیریم. فرض کنید فرآیند P3 نمونه ای از منبع نوع R2 را درخواست می‌کند. و چون فعلاً هیچ نمونه منبعی وجود ندارد، یال درخواست P3 →R2 به گراف اضافه می‌شود





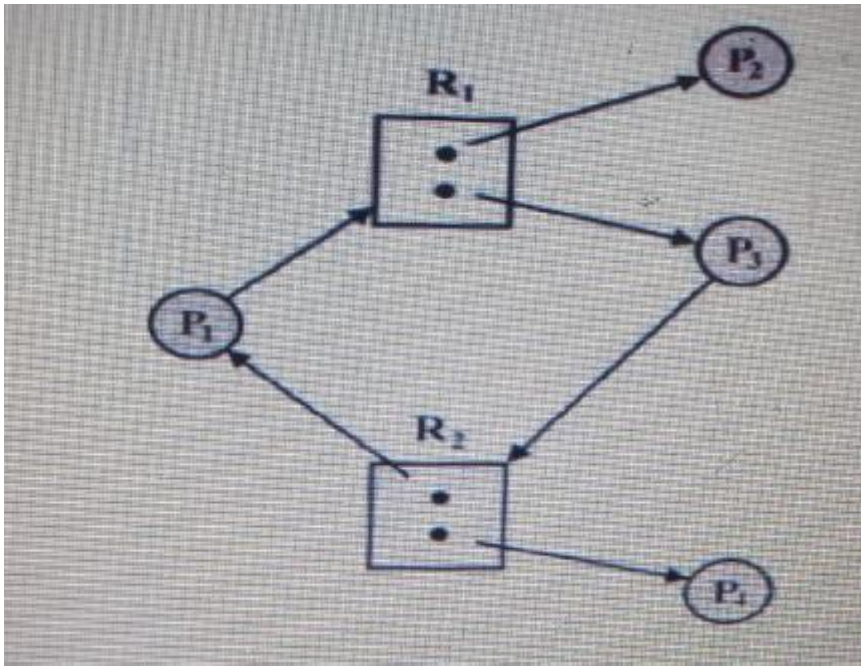
در این نقطه، دو چرخه‌ی کمینه سیستم وجود دارد.

P1 →R1 →P2 → R3 →P3 → R2 → P1

P2 →R3 →P3 →R2 →P2

فرایندهای P1,P2,P3 در بن بست قرار دارند. فرایند P2 منتظر منبع R3 است که این منبع در اختیار P3 است. فرایند P3 منتظر P1 یاP2 است تا منبع R2 را آزاد کنند. علاوه بر این، فرآیند P1 منتظر فرآیند P2 است تا منبع R1 را آزاد کند.

اکنون گراف تشخیص منبع به شکل زیر را در نظر بگیرید:



در این مثال یک چرخه وجود دارد:

P1 → R1 → P3 → R2 → P1

با وجود چرخه در این گراف، بن بستی وجود ندارد. فرایند P4 می‌تواند نمونه منبع R2 را آزاد کند. آن منبع می‌تواند به P3 تخصیص یابد. و چرخه از بین برود.

اگر گراف تخصیص منبع فاقد چرخه باشد، حالت بن بست وجود ندارد. اگر چرخه ای وجود داشته باشد، سیستم ممکن است در حالت بن بست باشد.