

سیستم عامل جلسه 8

بن بست ها (Dead locks)

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

شاید بهترین مفهوم از بن بست، از قانونی استنباط شود که قانون گذاران به تصویب رساندند. بخشی از قانون می گوید "وقتی دو قطار در یک تقاطع به هم نزدیک می شوند و هر دو باید کاملاً بايستند و هیچ کدام نباید حرکت کنند، مگر این که دیگری رفته باشد".

در این فصل، روش هایی را توصیف می کنیم که سیستم عامل می تواند برای جلوگیری از بن بست یا اداره کردن آن به کار گیرد. گرچه بعضی از برنامه های کاربری می توانند تشخیص دهند چه برنامه هایی دچار بن بست می شوند، ولی سیستم های عامل امکاناتی برای پیشگیری از بن بست ندارند و برنامه نویس باید تضمین کند که برنامه های آن ها فاقد بن بست باشد. با توجه به گرایش های فعلی؛ مثل تعداد زیاد فرایندها، برنامه های چند نخه (multi thread)، منابع زیاد در یک سیستم و تاکید بر سرور های پایگاه داده و فایل هایی با طول عمر زیاد به جای سیستم های 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 می شوند که هر کدام ممکن است شامل چند نمونه ی یکسان باشند

فضای حافظه، چرخه های پردازنده، فایل ها، دستگاه های IO (مثل چاپگر و گرداننده های DVD) از انواع منابع (Resource) هستند.

اگر فرایند نمونه ای از یک نوع منبع را درخواست کند، تخصیص هر نمونه از آن نوع، آن درخواست را برآورده می کند. اگر درخواست برآورده نشود، آن گاه نمونه ها 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 می کند،

این دو چاپگر ممکن است در یک دسته از منبع قرار گیرند. اما اگر یک چاپگر در طبقه ی نهم و چاپگر دیگر در طبقه ی همکف باشد، افرادی که در طبقه ی نهم قرار دارند می دانند که در هر دو چاپگر یکسان عمل نمی کنند و در نتیجه لازم است برای هر چاپگر دسته ی جداگانه ای از منبع تعریف کرد

هر فرایند قبل از به کارگیری منبعی باید آن را درخواست 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 را اختیار فرایند دیگری باشد)، فرایند درخواست کننده باید منتظر بماند تا منبع را در اختیار بگیرد

2. به کار گیری

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

3. آزاد کردن

فرایند منبع را آزاد می کند.

درخواست و آزاد سازی منابع، فراخوان های سیستم هستند. نمونه هایی از فراخوان های سیستم عبارتند از: `request()` و `release()` برای دستگاه ها، `open()` و `close()` برای فایل ها، `allocate()` و `free()` برای حافظه، درخواست و آزادی سازی 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 استفاده ی فرایند یا نخ از منبع تحت مدیریت هسته، سیستم عامل بررسی می کند تا مطمئن شود که فرایندی منبعی را درخواست کرده باشد و منبع به آن تخصیص داده شود، یک جدول سیستم، ثبت می کند که کدام منبع آزاد و کدام فرایند تخصیص یافته است. اگر فرایندی منبعی را درخواست کند که فعلاً به فرایند دیگری تخصیص یافته باشد می تواند 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 وسیله ی یک فرایند دیگر از آن مجموعه رخ خواهد داد.

منابع ممکن است فیزیکی باشند مثل چاپگرها، گرداننده های نوار، فضای حافظه، و چرخه های پردازنده، یا ممکن است ما

فایل ها، سمافورها، ناظر ها

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

این مثال بن بست را نشان می دهد 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 گرداننده ی DVD را در نظر بگیرید. فرض کنید فرایند P1 گرداننده ی 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 می دهد

مشخصات بن بست

در بن بست ، اجرای فرایندها خاتمه پیدا نمی کند، منابع سیستم به هم گره میخورند و از کارهای دیگر جلوگیری می شود. قبل از پرداختن به راه حل های اداره کردن مساله ی بن بست، در تعیین کننده ی بن بست ها را توصیف می کنیم.

شرایط ضروری

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

1. انحصار متقابل

حداقل یک منبع باید در حالت غیر اشتراکی نگهداری شود، یعنی در هر زمان فقط یک فرایند می تواند از آن منبع استفاده کند. اگر فرایند دیگری آن منبع را درخواست کند، فرایند درخواست کننده باید منتظر بماند تا آن منبع آزاد شود

2. نگهداری و انتظار

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

3. بدون قبضه کردن Non preemptive

منابع نمی توانند قبضه شوند، یعنی آزادسازی منبع به عهده فرایند است که آن را در اختیار دارد و پس از کامل کردن وظیفه ی خود، آن را آزاد می کند

4. انتظار چرخشی

باید مجموعه ای از فرایندهای منتظر $\{p_0, p_1, p_2, p_3, \dots\}$ وجود داشته باشند که p_0 منتظر منبعی باشد که در اختیار p_1 است و p_1 منتظر منبعی باشد که در اختیار p_2 است و به همین ترتیب p_{n-1} منتظر منبعی است که در اختیار p_n است و p_n منتظر منبعی است که در اختیار p_0 است. برای وقوع بن بست هر چهار شرط باید وجود داشته باشد.

گراف تخصیص منابع (system resource allocation graph)

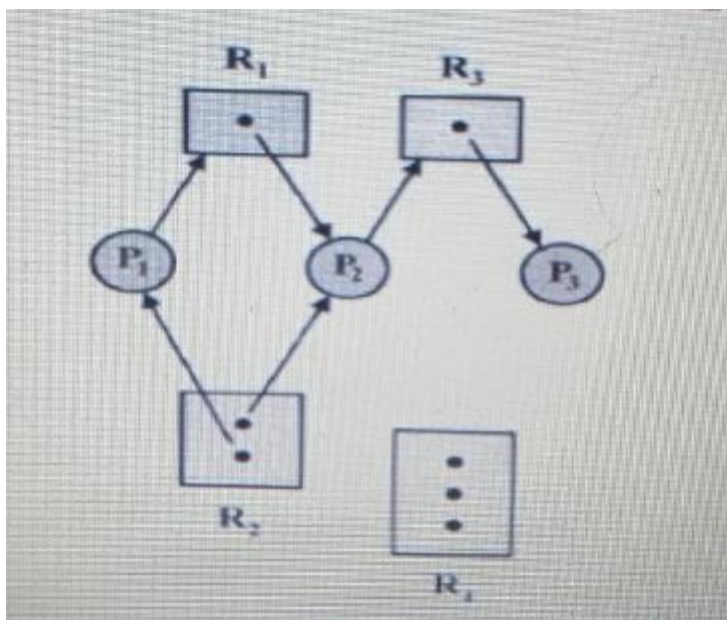
بن بست ها می توانند توسط گراف های جهت داری به نام گراف تخصیص منابع سیستم (system resource allocation graph)، دقیق تر تشریح شوند، این گراف شامل مجموعه ای از راس ها به نام V و مجموعه ای از یال ها به نام E است. مجموعه V به دو نوع گره های مختلف تقسیم می شود: مجموعه $P = \{P_1, P_2, P_3, \dots, P_n\}$ حاوی تمام فرایندهای فعال در سیستم است و مجموعه $R = \{R_1, R_2, R_3, \dots, R_m\}$ حاوی انواع منابع موجود در سیستم عامل است.

یک یال جهت دار از فرایند p_i به منبع R_j به صورت $P_i \rightarrow R_j$ نشان داده می شود. این علامت نشان می دهد که فرایند p_i نمونه ای منبع نوع R_j را درخواست کرده

است و منتظر آن منبع است. یک یال جهت دار از منبع نوع R_j به فرایند P_i به صورت $P_i \rightarrow R_j$ نمایش داده می شود و مشخص می کند که نمون از منبع نوع R_j به فرایند P_i تخصیص یافته است. یال جهت دار $P_i \rightarrow R_j$ یک یال درخواست (Request edge) نام دارد و یال جهت دار $R_j \rightarrow P_i$ یال تخصیص (Assignment edge) نام دارد.

در نمایش تصویری، هر فرایند P_i را با یک دایره و هر نوع منبع R_j را با یک مربع نمایش می دهیم . چون منبع R_j ممکن است چند نمونه داشته باشد و هر یک از نمونه ها را به صورت نقطه ای در مربع نمایش می دهیم. توجه کنید که یال درخواست ممکن است چند نمونه داشته باشد، هر یک از نمونه ها را به صورت نقطه ای در مربع نمایش می دهیم ، توجه کنید که یال درخواست فقط به مربع R_j اشاره می کند در حال که یال تخصیص باید به یکی از نقاط درون مربع نیز تخصیص یابد.

وقتی فرایند P_i نمونه ای از منبع نوع R_j درخواست می کند، یک یال درخواست به تخصیص منابع اضافه می شود، در صورتی که این درخواست انجام پذیر باشد، این یال درخواست فوراً به یک یال تخصیص تبدیل می شود، وقتی فرایندی به منبعی نیاز نداشته باشد، آن را آزاد می کند و در نتیجه یال تخصیص حذف می شود



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

مجموعه های E و R و P که عبارتند از :

$$P = \{P_1, P_2, P_3\}$$

$$R = \{R_1, R_2, R_3, R_4\}$$

$$E = \{P_1 \rightarrow R_1, P_2 \rightarrow R_3, R_1 \rightarrow P_2, R_2 \rightarrow P_2, R_2 \rightarrow P_1, R_3 \rightarrow P_3\}$$

نمونه ی منابع:

یک نمونه از منبع نوع R1

دو نمونه از منبع R2

یک نمونه از منبع R3

سه نمونه از منبع نوع R4

حالت های فرایند:

فرایند P1 نمونه از منبع نوع R1 را در اختیار دارد و منتظر نمونه ی از منبع نوع R1 است.

فرایند P2 یک نمونه از R1 و R2 را در اختیار دارد و منتظر نمونه ای از منبع نوع R3 است.

فرایند P3 یک نمونه از منبع R3 را در اختیار دارد.

با توجه به تعریف گراف تخصیص منابع، می توان 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.

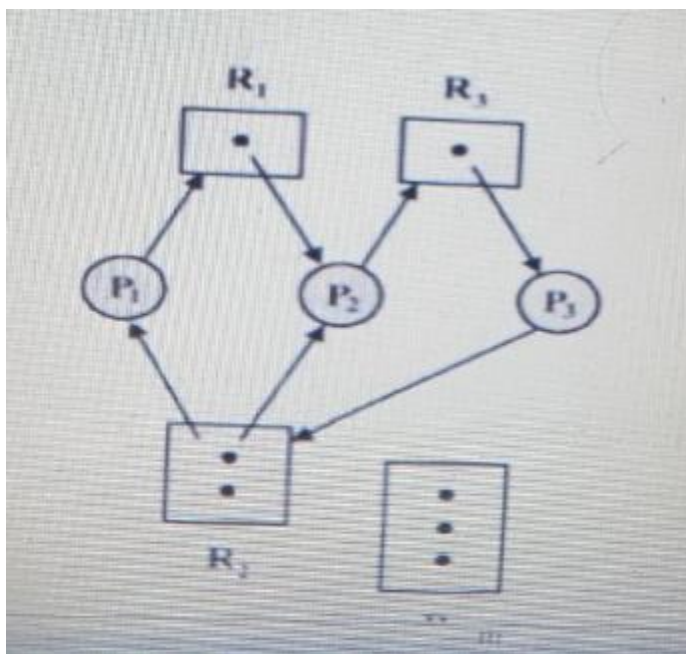
اگر هر نوع منبع دقیقاً یک نمونه داشته باشند. چرخه، نشان دهنده ی وجود بن بست است.

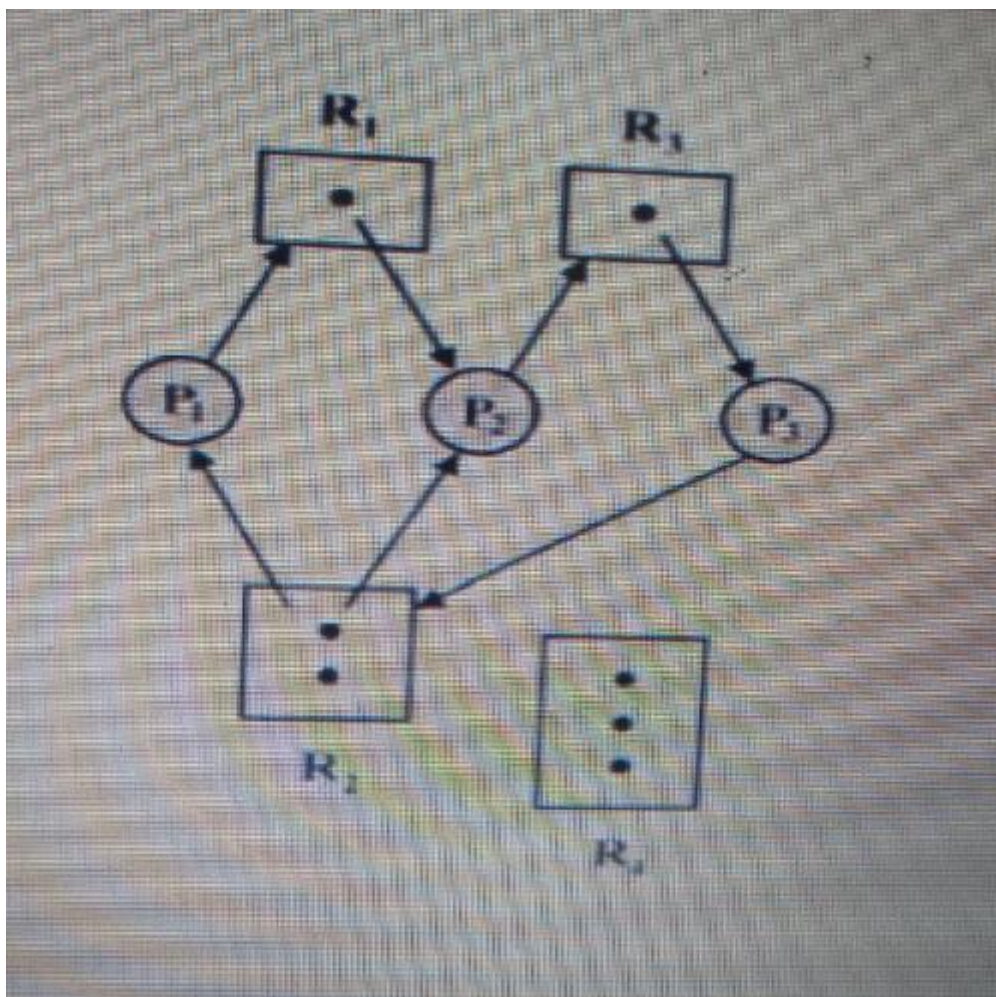
اگر چرخه، حاوی فقط یک مجموعه از انواع منبع باشد و هر نوع منبع شامل یک نمونه باشد، بن بست وجود دارد.

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 چرخه های را شرط لازم برای وجود بن بست است ولی شرط کافی نیست.

برای تشریح این مفهوم، گراف تخصیص منابع شکل قبلی را در نظر می گیریم.
 فرض کنید فرآیند P_3 نمونه ای از منبع نوع R_2 را درخواست می کند. و چون
 فعلا هیچ نمونه منبعی وجود ندارد، یال درخواست $P_3 \rightarrow R_2$ به گراف اضافه می
 شود





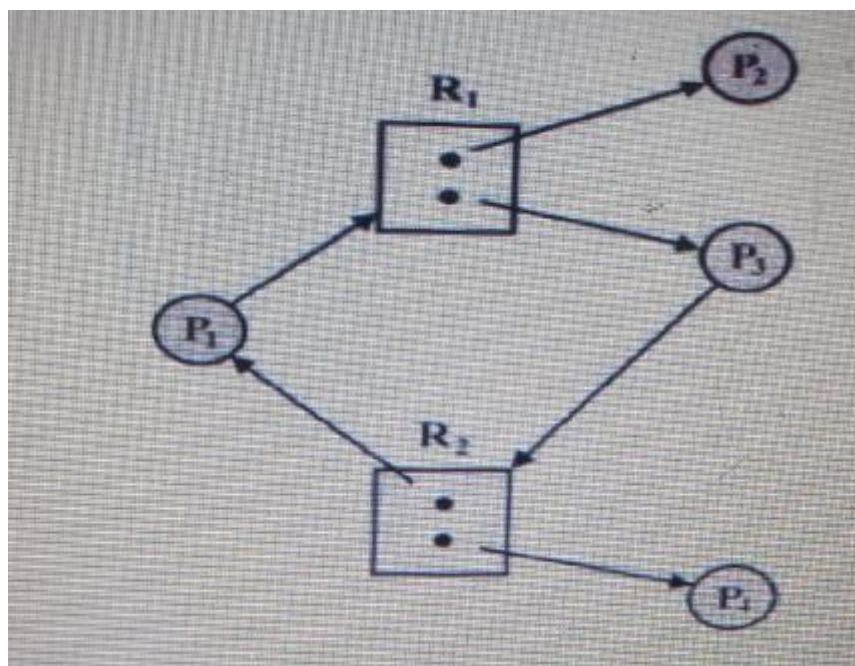
در این نقطه، دو چرخه ی کمینه سیستم وجود دارد.

$P1 \rightarrow R1 \rightarrow P2 \rightarrow R3 \rightarrow P3 \rightarrow R2 \rightarrow P1$

$P2 \rightarrow R3 \rightarrow P3 \rightarrow R2 \rightarrow P2$

فرایند های $P1, P2, P3$ در بن بست قرار دارند. فرایند $P2$ منتظر منبع $R3$ است که این منبع در اختیار $P3$ است. فرایند $P3$ منتظر $P1$ یا $P2$ است تا منبع $R2$ را آزاد کنند. علاوه بر این، فرایند $P1$ منتظر فرایند $P2$ 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:

$$P1 \rightarrow R1 \rightarrow P3 \rightarrow R2 \rightarrow P1$$

با وجود چرخه در این گراف، بن بست وجود ندارد. فرایند P4 می تواند نمونه منبع R2 را آزاد کند. آن منبع می تواند 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.