**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 دراختیار 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 گذازان به تصویب رساندند. بخشی از قانون می گوید "وقتی دو قطار در یک تقاطع به هم نزدیک می شوند و هر دو بایدکاملا باستند 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 توجهبه گزایش های فعلی؛ مثل تعداد زیاد فرایند ها، برنامه های چند نخی(multi thread، منابع زیاد در یک سیستم و تاکیدبر 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 فرایند های 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، ان درخواست را برآورده می کند. اگر درخواست برآورده نشود، ان 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 باشدو 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 چاپکر دسته ی 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 سیستمباشد.

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

**1.درخواست**

اگر درخواست نتواند فورا عملی شود(مثلا منبع ر 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(مثلا 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 سازیِ منابعی ک تحت مدیریت سیستم عامل نیستند. از طریق عملیات wait() و signal() بر روی سمافورها یا از طریق به درست آوردن و آزاد سازیقفل انحصار متقابل(mutex) صورت میگیرد. برای هر استفاده ی فرایند یا نخ از از منبع تحتِ مدیریتِ هسته، سیستم عامل بررسی می کند تا مطمین شود که فرایندی منبعی را درخاست 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 فعلا به فرایند دیگری تخصیص یافته باشدو می تواند به صف فرایندهای منتظر آن منبع اضافه شود.

مجموعه ای از فرایند ها 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 ان 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