# بخش اول: توضیح کد

* برای بحث اشتراک گذاری منابع دو الگوریتم HLP و NPP پیاده سازی شدند. برای مقدار دهی الگوریتم کافی است یکی از دو مقدار کلاس ResourceManagementAlgorithm انتخاب شود.

برای راحتی کار با کد های داده شده، هر کلاس در فایل جداگانه قرار گرفته شد که جداگانه توضیح داده خواهند شد:

* کلاس Job

برای پیاده سازی کلی سیستم، به دو تابع getRecourseWaiting و getRemainingSectionTime نیاز نشد!

این کلاس چندین تغییر جزئی دارد. برای مثال دو نوع اولویت تعیین شده که متغیر initial\_priority یا به اختصار ip اولویت اولیه سمافور را دارد و متغیر priority اولویت اصلی و دینامیکی را دارد که بسته به نوع الگوریتم استفاده شده می‌تواند تغییر کند.

تابع getResourceHeld آیدی سمافوری را برمی‌گرداند که درحال اشغال توسط scheduler است و تابع nextSectionJustStarted مشخص می‌کند آیا می‌خواهیم وارد سمافور جدید شویم یا نه.

* کلاس Task

تغییری نکرده

* کلاس Utils

یک Enum با نام ResourceManagementAlgorithm اضافه شده

* کلاس Taskset

تنها تغییر این کلاس اضافه کردن تابع build\_lowest\_priority\_semaphores برای ساختن متغیر lowest\_priority\_semaphores است. همانطور که از اسم متغیر پیداست، یک دیکشنری از سمافور ها را نگه می‌دارد که کمترین اولویت را دارند.

* کلاس Scheduler

تابع get\_semaphore\_priority اولویت سمافور ورودی را بر حسب الگوریتم زمانبندی برمیگرداند.

تابع get\_active\_jobs جاب هایی که تمام نشده (not completed yet) و همچنین از زمان آنها نگذشته باشد را برمیگرداند.

تابع build\_timeline مدت اجرای هر سمافور از هر جاب از هر تسک را نمایش میدهد.

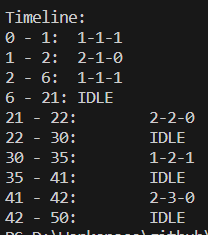
تابع tick بر حسب آنکه جاب فعلی در حال اجرا چه وضعیتی دارد (تمام شده و ...) سمافور ها را در صورت امکان اجرا می‌کند.

در الگوریتم HLP اولویت جاب برابر اولویت کمترین سمافور خواهد شد و در الگوریتم NPP اولویت جاب برابر صفر می‌شود.

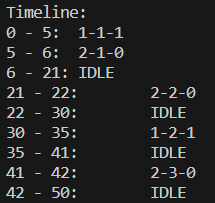
# بخش دوم: اجرا ها

برای مقایسه دو الگوریتم NPP و HLP از تسک کیس taskset\_2.json استفاده شده. به طور کلی الگوریتم HLP به دلیل تغییر اولویت به سمافور پیشین بهینه تر از NPP عمل میکند که صرفا اولویت را معادل 0 میگذارد. این امر در تست کسی گفته شده نشان داده شد. در تست کیس گفته شده تسک اول تسک دوم را بلاک کرده و مانع رسیدن آن به ددلاینش میشود.

اجرا با HLP:

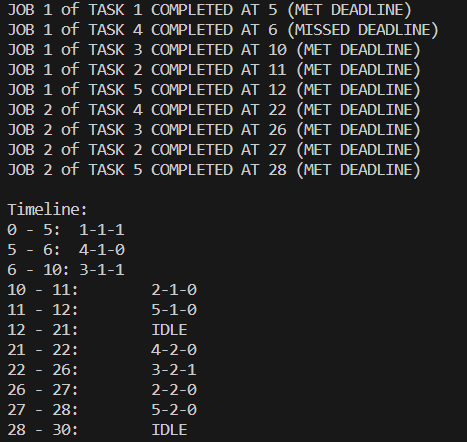


اجرا با NPP:

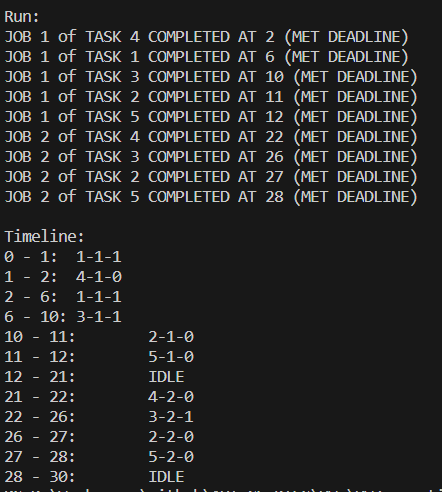


همچنین الگوریتم HLP تعداد کمتری سمافور را اصطلاحا بلاک کرده نسبت به NPP زیرا تسک هایی که اولویت بالاتری دارند بلاک نمیشوند بلکه اولویت آنها عوض می‌شود. برای توصیف دقیق تر تست کیس taskset\_3.json اجرا شده که در NPP موجب از دست رفتن ددلاین به دلیل بلاک شدن تسک با اولویت بالاتر میشود نشان داده شده:

اجرا با NPP:



اجرا با HLP:



# بخش سوک: مقایسه ها

* تاثیر offset

تغییر offset به طور کلی می‌تواند روی نحوه زمانبندی و حتی احتمال از دست رفتن ددلاین تاثیر گذار باشد زیرا ترتیب و زمان ورود تسک ها اهمیت زیادی دارد. ولی به طور کلی همانطور که بالاتر توضیح داده شد HLC نسبت به offset کمتر تحت تاثیر قرار میگیرد تا NPP به دلیل بهینگی بیشتر آن

* تعداد ناحیه بحرانی

در NPP کاهش یا افزایش تعداد ناحیه های بحرانی تاثیر زیادی روی عملکرد سیستم نمی‌گذارد. اما برای HLP هرچه تعداد ناحیه ها بیشتر شود تعداد بلاک ها هم به صورت متناظر زیاد می‌شود به طوری که به NPP میل می‌کند.

* تعداد تسک ها

طبیعتا با افزایش تعداد تسک ها، تعداد بلاک ها هم برای هر دو الگوریتم HLP و NPP بیشتر می‌شود. این نسبت متناسب با مشخصات تسک ها می‌تواند کم و زیاد باشد اما به طور کلی اینطور است.