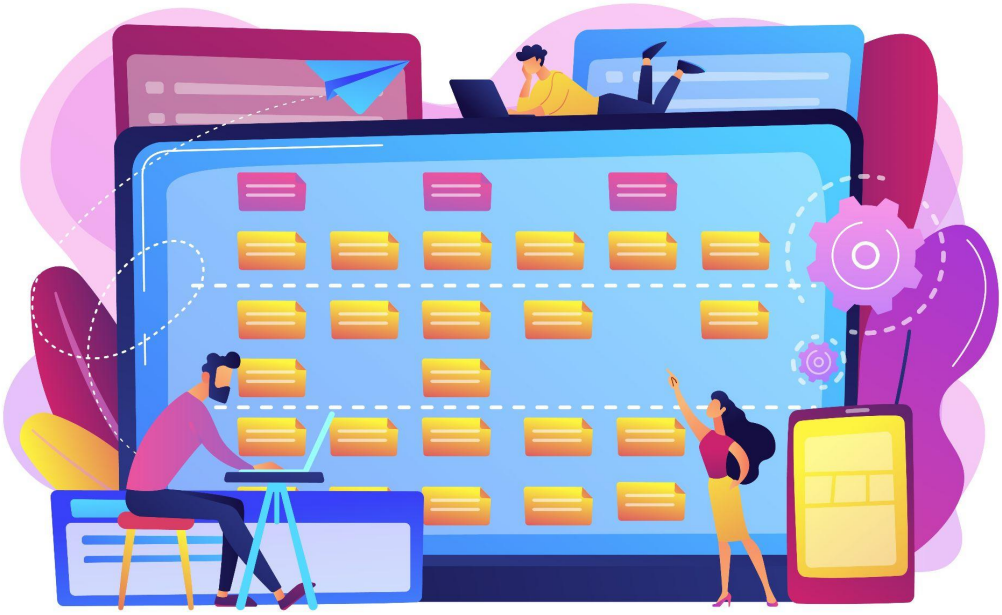
# پروژه دوم درس سیستم‌های‌عامل

**Scheduler**, Multi-Processor System Simulator



## شرح پروژه

قصد داریم در این پروژه یک Scheduler را برای یک سیستم با چهار پردازنده پیاده‌سازی کنیم. در این شبیه‌ساز از یک نخ برای هر پردازنده و یک نخ جداگانه که با چهار پردازنده در ارتباط است استفاده می‌شود.
وظیفه نخ مشترک **چاپ خروجی برنامه** است که شامل اطلاعات مربوط به وظیفه انجام شده و پردازنده‌ای که آن را انجام داده، می‌شود.

در سیستم همچنین منابعی (Resources) وجود دارد که برای انجام وظیفه‌ها به آن‌ها نیاز داریم. تعداد این منابع محدود است و وظیفه‌های مختلف بسته به نوعشان به منابع متفاوتی نیاز دارند. اگر یک وظیفه در اول صف اولویت قرار گیرد ولی منابع مورد نیاز برای آن موجود نباشد، از صف Ready خارج شده و وارد صف Waiting می‌شود.

# مفاهیم پروژه

---

## منابع

در سیستم سه نوع منبع (R1, R2, R3) وجود دارند که هنگام شروع برنامه تعداد موجود از هر کدامِ آن‌ها در سیستم به شما داده می‌شود.

## وظایف

در این سیستم سه نوع وظیفه (X, Y, Z) وجود دارد. اولویت میان این وظایف به شرح زیر است:

| Task | Priority |
|---|---|
| Z | 1 (the most) |
| Y | 2 |
| X | 3 |

همچنین نوع و تعداد منابع مورد نیاز آن‌ها ثابت بوده و به صورت زیر می‌باشد:

- وظیفه X به منابع R1 و R2 نیاز دارد.
- وظیفه Y به منابع R2 و R3 نیاز دارد.
- وظیفه Z به منابع R1 و R3 نیاز دارد.

برای هر وظیفه مدت زمان مورد نیاز برای اجرای آن داده می‌شود. همچنین باید در ساختار وظیفه فیلدی برای **ذخیره وضعیت وظیفه** در نظر بگیرید که نشان‌دهنده‌ی State آن در سیستم است (Ready/Waiting/Running).

همچنین باید فیلدی مربوط به **میزان زمانی که تسک بر روی پردازنده** قرار گرفته است نیز تعریف کنید.

## الگوریتم‌های زمان‌بندی

الگوریتم‌هایی که باید پیاده‌سازی شوند:

- Shortest Job First
- First Come First Service
- Round Robin

الگوریتم‌هایی که پیاده‌سازی آن‌ها نمره‌ی اضافه دارد:

- Highest-Response-Ratio-Next/HRRN
- Multilevel Feedback Queue

### صف Ready

این صف مربوط به وظیفه‌هایی میشوند که آماده اجرا هستند و ترتیب آن‌ها با توجه به الگوریتم‌های زمان‌بندی مطرح شده مشخص می‌گردد. فقط یک صف اولویت در سیستم وجود دارد.

### صف Waiting

این صف مربوط به وظیفه‌هایی می‌شود که امکان اجرای آن‌ها وجود دارد، ولی منابع مورد نیاز آن‌ها موجود **نیست**.

مثلاً هنگامی که یک پردازنده وظیفه‌های از صف اولویت انتخاب می‌کند ولی منابع آن موجود نیست، سیستم این وظیفه را از صف اولویت خارج کرده و در صف انتظار قرار می‌دهد.

برای جلوگیری از Starvation، باید راه‌حلی برای برگرداندن وظیفه‌ها به صف اولویت در نظر گرفته شود. در نتیجه لازم است الگوریتمی برای مرتب کردن این صف با توجه به منابع آزاد سیستم و به دست آوردن بهترین بهره‌وری از پردازنده‌ها پیاده‌سازی شود.

در صورتی که وظیفه‌ای از صف انتظار به صف اولویت برگردد و مدت زیادی در صف انتظار قرار گرفته بوده یا زمان باقی مانده اجرای آن نسبت به بقیه وظیفه‌های روی پردازنده‌ها کم باشد، با توجه به الگوریتم زمان‌بندی یا باید اولویت آن افزایش یابد یا در اول صف قرار بگیرد یا جایش با یکی از وظیفه‌های در حال اجرا عوض شود.

### زمان

هر واحد زمان را می‌توانید یک دور در حلقه اصلی برنامه‌تان در نظر بگیرید.

### همگام‌سازی (Synchronization)

در شبیه‌ساز فقط یک صف Ready و Waiting وجود دارد (در میان چهار پردازنده مشترک هستند) پس باید از وقوع race condition برای نخ‌های پردازنده هنگام دسترسی به آنها جلوگیری کرد. برای این کار از mutex استفاده کنید.
برای سایر منابع مشترک در سیستم نیز (مانند منابع مشترک بین نخ کنترل که وظیفه چاپ وضعیت کلی سیستم را دارند، پردازنده‌ها، وظیفه‌های در حال اجرا، صف‌ها و نخ‌های پردازنده) **در صورت امکان وقوع race condition** از mutex استفاده کنید.

## فرمت ورودی و خروجی

---

### ورودی

- در خط اول به ترتیب (از چپ به راست)، تعداد منابع موجود در سیستم برای R1, R2, R3 قرار می‌گیرد.
- در خط دوم، تعداد وظیفه‌هایی که قرار است زمان‌بندی شوند وارد می‌گردد.
- از خط سوم به بعد، وظیفه‌ها به شکل [Task_Name, Task_Type, Task_Duration] وارد می‌شوند، به عنوان مثال:

```
T1    Y    3
T2    X    6
T3    X    5
T4    Z    1
```

### خروجی

- بعد از هر واحد زمان، وضعیت دو صف موجود در سیستم، تعداد منابع موجود و وضعیت هر پردازنده (شامل وظیفه در حال اجرا بر روی آن) باید نمایش داده شود.
- در صورتی که امکان قرارگیری تسکی بر روی پردازنده وجود نداشته باشد، روبه‌روی آن Idle به معنی بیکار بودن قرار می‌گیرد.

مثلاً:

```
R1: 0    R2: 2    R3: 0
Priority queue: [T2-T1-T3]
Waiting queue: [T6]
CPU1: [T5]
CPU2: [T4]
CPU3: Idle
CPU4: [T7]
```

# پروژه دوم درس سیستم‌های‌عامل

**Scheduler**, Multi-Processor System Simulator

## توضیحات تکمیلی

---

1) پروژه در گروه‌های 2 نفره قابل انجام است (آپلود توسط یکی از اعضا کافی‌ست).
2) استفاده از زبان‌های Java, Python, C, CPP مجاز است (در صورت استفاده از C/C++ از makefile استفاده شود).
3) مراحل پیاده‌سازی و نحوه‌ی اجرای برنامه‌ی خود را حتماً در فایل readme.md **به صورت کامل** توضیح دهید.
4) هنگام تحویل، *هر دو عضو* گروه باید **تسلط کامل** داشته باشند.
5) یک نمونه ورودی و خروجی از برنامه خود را در قالب دو فایل in.txt و out.txt ذخیره داشته باشید.
6) فایل نهایی (شامل کد، فایل readme.md، فایل in.txt و فایل out.txt) را به‌فرمت "Scheduler_<Student.IDs>_<Student.names>.zip" در Vu بارگذاری کنید.
7) در صورت مشاهده هرگونه شباهت میان گروه‌ها، ***نمره 100-*** به هر دو گروه داده می‌شود.

---

**مهلت تحویل:** 13 بهمن ماه 1402 خورشیدی.

"موفق باشید"