

# ProcessManager

---

**Augustin MARMUS | Clément DOUMERGUE**

Ce programme est une simple simulation de l'ordonnancement des processus et de l'allocation de la mémoire au sein du Système d'Exploitation. Cette simulation se présente sous la forme d'une interface WEB. Elle est faite en Angular 7.

## Compilation

---

Ce dépôt fournit un dockerfile permettant de compiler simplement:

```
docker build --rm . -t process-manager:latest
```

Lancer la simulation:

```
docker run --rm -i -t -p 8000:80 process-manager:latest
```

Vous pourrez trouver la simulation à l'adresse <http://localhost:8000>

Ou plus simplement avec docker-compose:

```
docker-compose up
```

## Manuel d'utilisation

---

Vous pouvez paramétrer plusieurs facteurs dans l'onglet settings (nombre de cœurs, l'algorithme d'ordonnancement, la taille de la mémoire, l'algorithme de d'allocation).

Vous pouvez ajouter/supprimer/éditer des processus dans le menu.

Le player en haut du menu permet de lancer la simulation (Jouer, Pause, Pas à pas, Reset).

## Implémentation

---

### Structure de données

Les Processus sont la principale structure de données, ils sont composés de Threads et de Pages.

Process

```
class Process {

    private static nextId: number = 0;

    name: string = 'Process ' + Process.nextId;
    id: number = ++(Process.nextId);
    priority: number = 1;
    computed: number = 0;
    ioed: number = 0;
    threads: Thread[] = new Array<Thread>(new Thread(this));
    pages: Page[] = new Array<Page>();
}
```

Ils contiennent les pages et les threads dont ils sont composé:

Thread

```
class Thread {
    private parentProcess: Process;
    state: Thread.STATE;
    remainingTime: number;
    inactivityTimeStamp: number;
}
```

Page

```
class Page {
    private parentProcess: Process;
    state: Page.STATE;
    allocatedTimeStamp: number = -1;
}
```

## Interfaces

Les différents algorithmes d'ordonnancement sont implémentés à l'aide d'interfaces qui fournissent des hooks ou les différents algorithmes peuvent agir sur la simulation:

Scheduler

```
interface Scheduler {
    onTimeUnitStart(processes: Process[], nbCores: number, tick: number): void;
    onTimeUnitEnd(processes: Process[], nbCores: number, tick: number): void;

    name: string;
    description: string;
}
```

Allocator

```
interface Allocator {  
    onTimeUnitStart(processes: Process[], memory: Page[], tick: number): void;  
    onTimeUnitEnd(processes: Process[], memory: Page[], tick: number): void;  
  
    name: string;  
    description: string;  
}
```

## Décisions d'implémentation

---

L'ordonnancement est préemptif: les processus en attente d'I/O ne bloquent pas les cœurs. La décision de choix entre opération d'I/O et de calcul est aléatoire lorsque le processus contient les deux. L'algorithme par priorité contient un système anti-famine: plus le processus a été inactif plus sa priorité augmente. La taille des pages est fixée à une instruction.