# Info0403 (OS)

## Rapport de projet

*Dupont Corentin*
*Lacroix Owen*





*2020-2021*

# Table des matières

Table des matières ..... 2
**Le projet :** ..... 3
**La réalisation :** ..... 3
**Exemple :** ..... 4
**Pisete d'améliorations :** ..... 6

# Le projet :

Le projet à réaliser est une élection de processus. Nous devons réaliser un code permettant de générer un nombre de processus et de leur attribuer un id aléatoire (avec la valeur 1 en minimum). Le critère d'élection est l'id du processus. En effet le processus élu est le processus ayant le plus petit id. Pendant l'élection, les processus s'échange un message où ils incrémentent la valeur "hop". Les processus sont tous candidats par défaut, et deviennent battus s'ils ne sont pas élus. Le processus élu doit alors faire remonter son id à la racine, qui le transmettra ensuite à tous les autres processus.

# La réalisation :

Nous avons créé une structure Process, définie par un état (Candidat (défaut), Battu; Elu) qui est une énumération, un id et un pid. L'id du processus est défini aléatoirement. Le pid lui est récupéré à un fork et est assigné au processus.
Nos processus ainsi créés sont stockés dans un tableau.

**Le processus d'élection se déroule de la sorte :**

- Nous créons les processus participant à l'élection. Nous définissons aléatoirement leur id avec une fonction dédiée à la génération d'id, et leur attribue le pid du fork dans lequel le processus est créé.

- Nous lançons l'élection. Nous utilisons une fonction dédiée pour récupérer la valeur d'id minimum puis nous parcouron sensuite le tableau pour vérifier pour chaque

processus si l'id associé est égal au minimum avec une autre fonction.

- Si l'id du processus est supérieur au minimum, le processus est déclaré battu. En revanche, s'il est égal, le processus candidat reste en course. Pour chaque processus ayant l'id minimum, on incrémente un compteur. Si le compteur est supérieur à 1, nous relançons une élection, car cela signifie que plusieurs processus ont l'id minimum. Ses processus se voient réattribuer un nouvel id, et une procédure d'élection est relancée avec seulement les candidats encore dans la course, et ainsi de suite jusqu'à ce que le compteur soit égal à 1. Une fois, cela atteint alors le processus restant est déclaré gagnant.

## Exemple :

Dans l'affichage en console nous avons décidés de ne pas afficher les processus battus pour ne pas surcharger et faciliter la lecture.

### Exemple avec 1 tour :

```
------Candidats en jeu : -----
-Processus #0 (id = 1, PID = 74125) : Candidat
-Processus #1 (id = 2, PID = 74126) : Candidat
-Processus #2 (id = 5, PID = 74127) : Candidat
-Processus #3 (id = 3, PID = 74128) : Candidat
-Processus #4 (id = 2, PID = 74129) : Candidat
-----Election : -----

-------------------------------Gagnant-----------------------------------
Processus #0 (id = 1, PID = 74125) : Elu
-------------------------------------------------------------------------
```

## Exemple avec plusieurs tours :

```
------Candidats en jeu : -----
-Processus #0 (id = 1, PID = 75561) : Candidat
-Processus #1 (id = 5, PID = 75562) : Candidat
-Processus #2 (id = 3, PID = 75563) : Candidat
-Processus #3 (id = 4, PID = 75564) : Candidat
-Processus #4 (id = 1, PID = 75565) : Candidat
-----Election : -----
=> Nouvelle election :
-Processus #0 (id = 1, PID = 75561) : Candidat
-Processus #4 (id = 2, PID = 75565) : Candidat

-------------------------------Gagnant-----------------------
Processus #0 (id = 1, PID = 75561) : Elu
-------------------------------------------------------------
```

# Pisete d'améliorations :

Nous n'avons pas fait la gestion des signaux, et nous n'avons pas réussi à faire l'implémentation du système de messages. Nous avons donc au moins 2 pistes pour améliorer notre code. Pour finir, nous avons tout de même essayé de réaliser un système d'élection de processus certes différente de la consigne, mais fonctionnel.