Rapport de Projet : Générateur de chronogrammes d'ordonnancement

Département IMA : Informatique Microélectronique Automatique

Élève : Samy BELHOUACHI

Tuteurs : Xavier REDON
Thomas VANTROYS

Table des matières

INTRODUCTION

Dans le cadre de ma quatrième année dans le département d'Informatique Microélectronique et Automatique de l'école Polytech Lille, j'ai été amené à participer au développement du projet « Générateur de chronogrammes d'ordonnancement ». Ce projet fait appel à des notions vues en cours, mais celui-ci m'offrait aussi la capacité de développer de nouvelles compétences.

Le but de ce projet est d'avoir une application web permettant de générer des chronogrammes d'ordonnancement. Si nous prenons le cas d'un ordinateur multi-programmé qui possède plusieurs processus en concurrence pour l'acquisition de temps processeur. A chaque fois que deux processus ou plus sont dans cette situation c'est-à-dire en état prêt au même moment, alors l'ordonnanceur utilise un algorithme d'ordonnancement. Ces algorithmes seront détaillés dans la seconde partie de ce rapport.

Dans ce rapport, je m'attarderai sur le cahier des charges. Puis je présenterai le travail réalisé. Ensuite, dans une troisième partie je réaliserai quelques tests de l'application Web. Enfin dans une dernière partie, j'expliquerai les difficultés que j'ai rencontré dans ce projet, puis je présenterai les différents bugs existants et j'essaierai de prendre du recul sur la globalité du projet.

I/ PRÉSENTATION DU CAHIER DES CHARGES

Le cahier des charges de ce projet est composé de différentes parties : la partie interface WEB, la partie affichage et la partie variante des algorithmes d'ordonnancement.

- L'interface utilisateur doit être de type Web en HTML5. De plus il est possible d'utiliser des bibliothèques JavaScript disponibles sur internet.

- Dans l'interface, les éléments suivants doivent être spécifiés par l'utilisateur :
  - Son type d'ordonnancement à utiliser
  - Son nombre de ressources disponibles
  - Son nombre de processus
  - Et pour chaque processus il faut préciser son PID, sa date de création, sa priorité et ses événements

- Il faudrait que les événements d'un processus soient saisis comme une liste d'opérations.

- Chaque opération possède une date de départ qui se comprend, non en temps absolu, mais en temps d'exécution du processus. Les opérations peuvent être du type « appropriation » de ressource, « libération » de ressource ou « arrêt » du processus. Pour l'appropriation ou la libération de ressources, l'utilisateur doit pouvoir préciser le numéro de cette ressource.

- Une fois tous les éléments précisés, la simulation d'ordonnancement est lancée directement sur le navigateur en JavaScript. La simulation se termine si tous les processus s'arrêtent ou si un interblocage se produit. Le chronogramme est alors affiché, il faut soigner cet affichage.

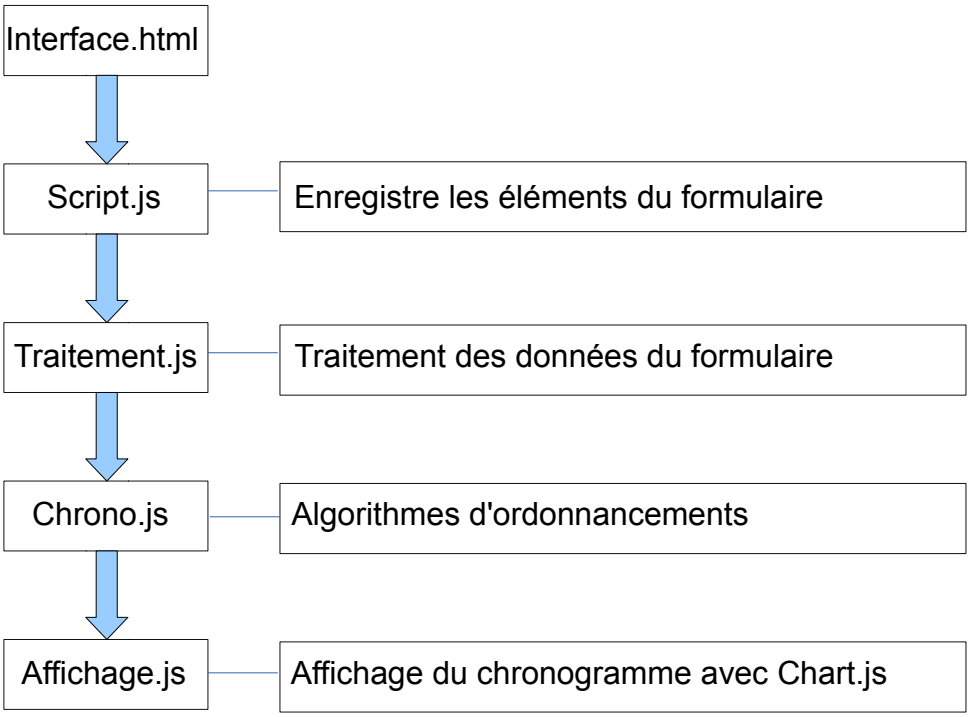
- L'autre aspect du projet consiste à étudier les différentes variantes des algorithmes d'ordonnancement. Nous nous attarderons un peu plus sur la gestion des ressources. Puis nous afficherons la variante.

II/ PRÉSENTATION DU TRAVAIL

Dans cette partie, l'explication de la structure de l'interface graphique sera effectuée. Ainsi que le cheminement de mon travail pour répondre au mieux au cahier des charges.

II.1/ Structure de l'interface Web

L'objectif de cette interface Web est de récupérer les données entrées par l'utilisateur permettant ainsi la création du chronogramme d'ordonnancement. Cette interface est composée d'un fichier de type HTML comprenant la première partie du formulaire, d'un code CSS et de 4 fichiers JavaScript. L'explication de l'utilité de ces différents fichiers sera réalisée par la suite.



II.2/ Page d'accueil de l'interface Web

Le langage HTML (HyperText Markup Language) est un langage de balisage conçu pour créer des pages web. Ce langage permet de structurer logiquement et sémantiquement et de mettre en forme le contenu des pages web, d'inclure des formulaires de saisies, des programmes informatiques et ressources multimédias. Dans le

cas de l'interface graphique j'ai mis les éléments suivants dans le formulaire de la page d'accueil :

- Un groupe de boutons radio facilitant le choix de l'algorithme
- Un champ « input » permettant de choisir le nombre de processus
- Un second champ « input » permettant cette fois-ci de choisir le nombre de ressources
- Et enfin un bouton « valider », pour valider les différentes informations entrées par l'utilisateur.

Une fois que tous ces éléments ont été entré par l'utilisateur, nous avons la suite du formulaire qui s'affiche.

II.3/ Suite du formulaire rendu dynamique

Pour rendre le formulaire dynamique, nous utilisons le JavaScript car celui-ci est un langage employé dans les pages Web interactives. Ce langage sert généralement à contrôler les données saisies dans des formulaires HTML, ou à interagir avec celui-ci, il est aussi utilisé pour réaliser des services dynamiques.

Comme nous l'avons expliqué précédemment, nous avons la première partie du formulaire qui s'affiche sur la page d'accueil. Et c'est une fois que l'utilisateur valide ses différents choix que la seconde partie du formulaire qui est dynamique s'affiche. Ainsi pour rendre dynamique le formulaire, il a fallu créer un fichier JavaScript qui permet de rendre le formulaire dynamique.

Dans le fichier "interface.html", nous avons donc une première partie du formulaire. Dans cette première partie, nous demandons à l'utilisateur de sélectionner le type d'ordonnancement, le nombre de processus et le nombre de ressources. Une fois toutes ces données entrées, l'utilisateur valide son choix. Une fois ce choix effectué, j'avais eu l'idée de masquer les données précédentes. Sachant que je voulais que le code soit dynamique, j'ai donc décidé de créer un fichier "script.js" me permettant de gérer la partie dynamique du formulaire.

Après avoir validé les différents éléments choisis, nous sommes mené vers une interface des processus. Ensuite, le nombre de processus choisi est récupéré par une fonction jquery, celle-ci ajoute le nombre de champs nécessaires.

Dans cette interface des processus, nous avons le nombre de processus que nous avons choisi. Ainsi, pour chaque processus nous avons :

- Un champ pour le PID
- Un champ pour la date de création

- Un champ pour la priorité

- Un champ « input » permettant d'entrer le nombre d'événements. J'ai donc ajouté un bouton permettant la validation de ce nombre choisi. Une fois le click enclenché nous avons le nombre d'événements qui s'affiche.

- Ces événements sont composés d'un temps, d'un type d'opérations et du choix d'une ressource

Une fois tous les champs remplis par l'utilisateur, nous devons cliquer sur le bouton « envoyer ». Après avoir cliqué nous récupérons toutes les données qui permettent le bon fonctionnement des différents algorithmes d'ordonnancement.

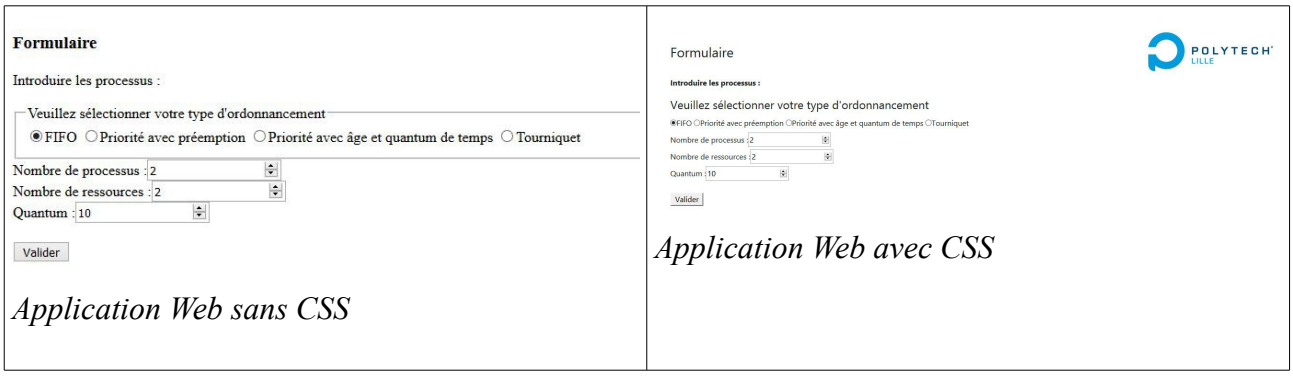
II.4/ Ajout et changements de cette interface

Après avoir pris un peu de recul, j'ai pensé a ajouté un « input » permettant d'entrer la valeur du quantum. Celui-ci est utile pour seulement deux types d'ordonnancements : le tourniquet et priorité avec âge et quantum de temps. Si par exemple nous mettons une valeur dans le quantum de temps et que nous choisissons un type d'ordonnancement qui n'a pas besoin de quantum cela n'affectera pas le déroulement de cette ordonnancement.

Ensuite, après avoir réalisé ma variante, j'ai décidé de créer un groupe de boutons radio permettant de choisir le type de gestion de ressources que nous voulons.

II.5/ Mise en place du CSS

L'interface web a été mise en place par l'intermédiaire de différents langages de programmation dont le CSS. J'ai utilisé le code HTML pour la mise en page simple de l'application. Et j'ai utilisé le CSS pour un apport visuelle et esthétique de la page web.



II.6/ Affichage texte des chronogrammes

J'ai aussi les différentes étapes de l'ordonnancement qui s'affiche sur la page ainsi j'ai décidé d'avoir la possibilité d'avoir un affichage texte pour pouvoir comparer avec l'affichage des chronogrammes. Cette affichage permet de voir le numéro des processus ainsi que le temps, les différentes opérations, et leur statut.

II.7/ Choix de l'outil pour l'affichage des chronogrammes

Au commencement de mon projet, je n'avais pas pensé à utiliser la bibliothèque « Chart.js ». J'avais d'abord pensé à utiliser WaveDrom qui était assez limité, pour le traçage des différents processus. J'ai donc cherché différentes solutions pour pouvoir avoir un affichage correcte. J'avais donc les options suivantes :

- PlotKit
- JS Charts
- Flot

J'ai préféré utiliser la librairie Charts.js qui était très bien documentée, avec de nombreux exemples. De plus j'ai donc tracé des chronogrammes manuellement pour voir si cela pouvait répondre aux différentes tâches du cahier des charges, le résultat était assez esthétique.

II.8/ Affichage et légende des chronogrammes

Concernant la gestion de l'affichage, il y a l'affichage des légendes, du chronogramme et des instants de capture des ressources ainsi que leur relâchement. Pour les instants de capture des ressources et leur relâchement j'ai choisi d'utiliser deux formes différentes

De plus, j'ai pris un certain temps pour la prise en main de cette bibliothèque, j'ai d'abord commencé par réaliser quelques exemples et étudier des exemples déjà existants. Cette étape fut essentielle pour que je comprenne comment il fallait introduire la légende, mais aussi pour pouvoir choisir l'aspect de visualisation pour le type d'opération, j'ai choisi différentes formes permettant de visualiser cela.

Au départ, on n'affichait seulement 5 processus. Sachant que nous avons dynamisé l'interface, il faut aussi rendre l'affichage dynamique. C'est-à-dire qu'il faut que les légendes s'affichent dynamiquement. Et que le cadrage et l'utilisation des différents espaces sur le graphique se fassent automatiquement.

Une fois ces parties modifiées concernant l'affichage, j'ai réalisé quelques tests.

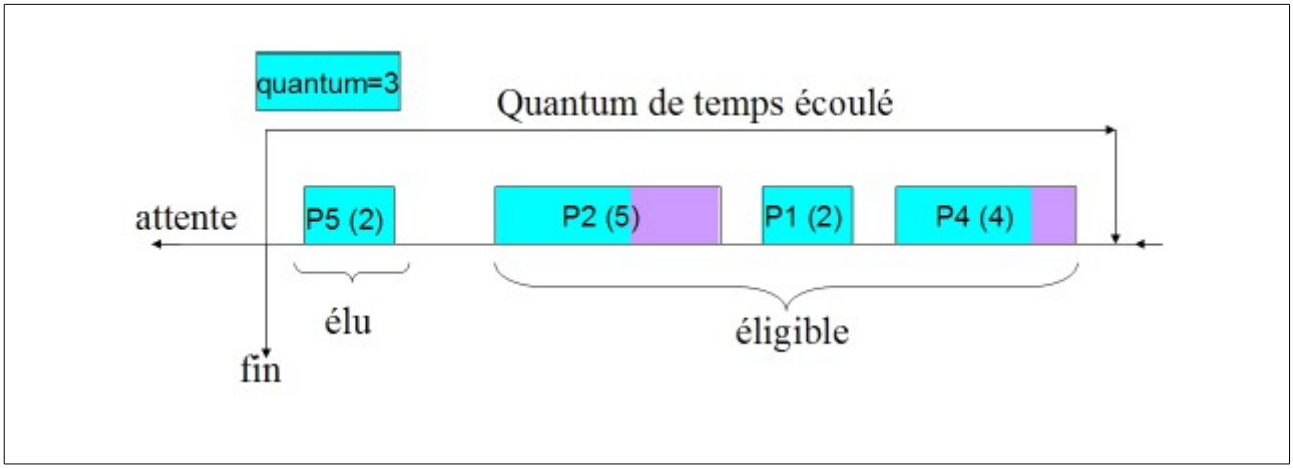
Ainsi, j'ai pensé à randomiser les couleurs pour ne pas avoir le problème d'avoir les mêmes couleurs pour des processus différents.

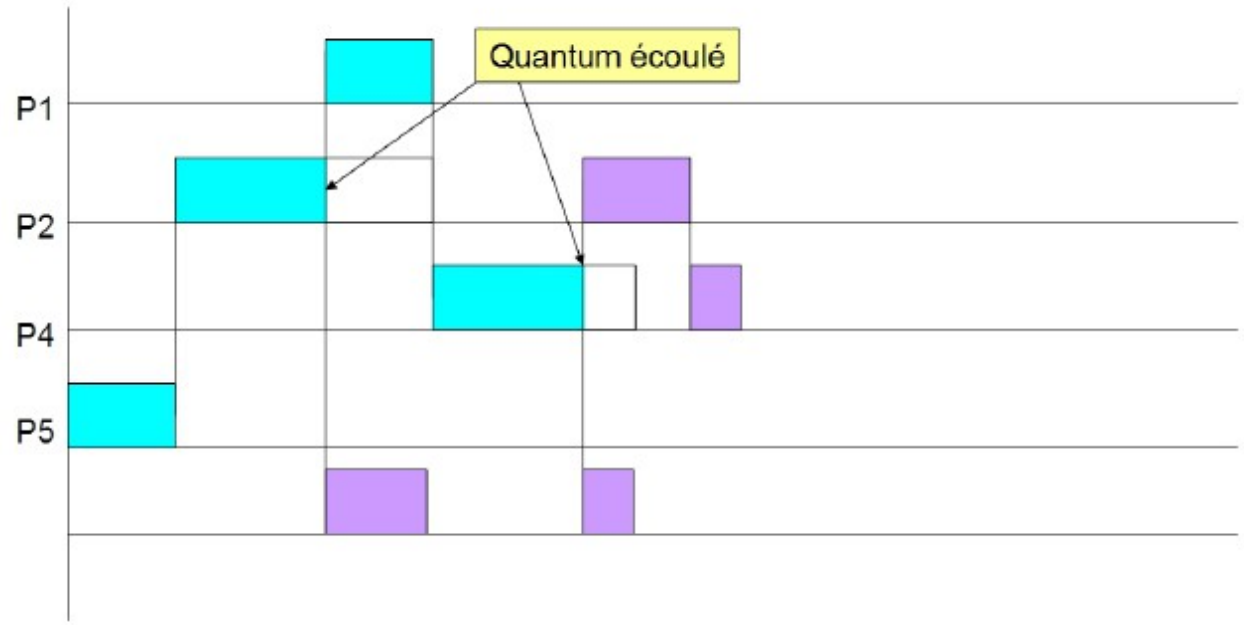
II.9/ Les différents types d'ordonnancements

Dans l'interface Web, l'utilisateur a le choix entre quatre types d'ordonnancement le FIFO, le Tourniquet, le priorité avec préemption et l'ordonnancement priorité âge et quantum de temps. Voici le rôle des différents types d'ordonnancements sont :

- **FIFO :** FIFO signifie First In, First Out. Cet ordonnancement fonctionne de la manière suivante premier entré, premier sorti.

- **Tourniquet :** Le tourniquet fonctionne de la manière suivante, chaque processus se voit attribuer un intervalle de temps appélé « quantum » pendant lequel il est autorisé à s'exécuter. Une fois son « quantum » terminé, celui-ci s'arrête pour laisser le tour à un autre processus. Et s'il lui reste encore du temps, ce processus repart en queue de liste pour attendre son prochain tour.

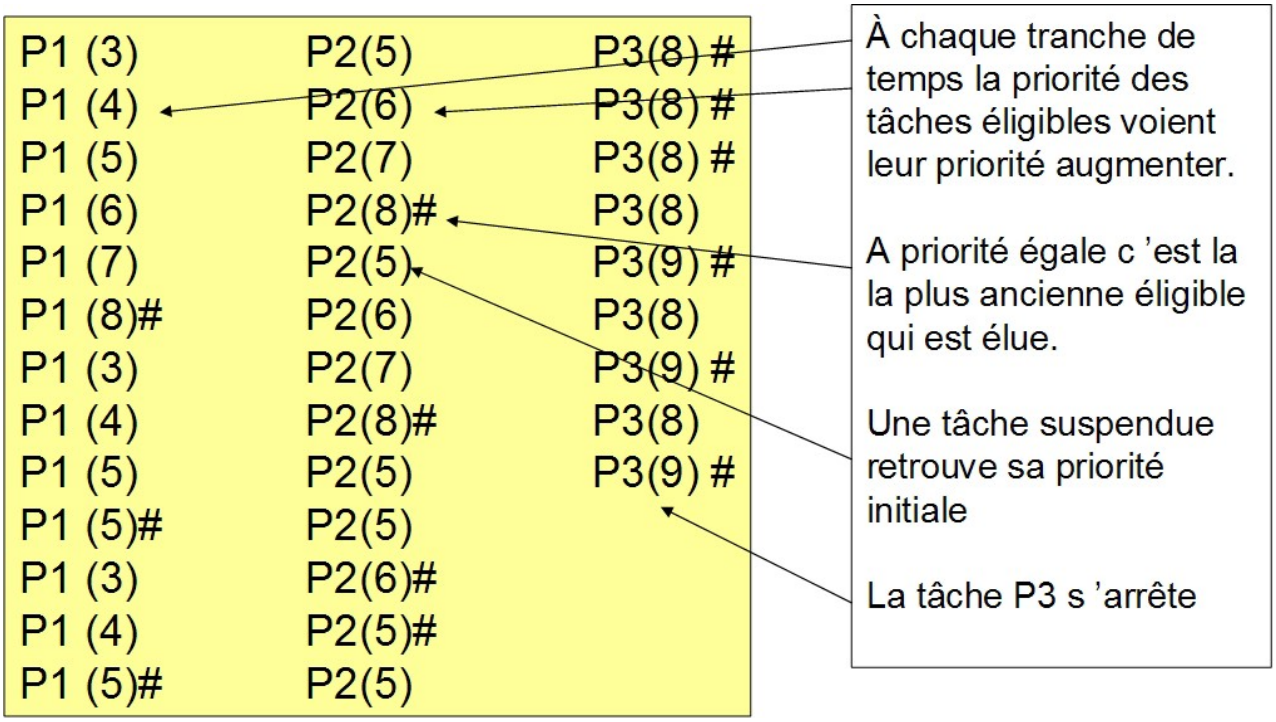




Fonctionnement du Tourniquet (Source cours de M. Vantroys)

- **Priorité avec préemption :** Ce type d'ordonnancement fonctionne de la manière suivante, chaque processus détient un niveau de priorité, et si le processus qui arrive dans la file d'attente est plus prioritaire que celui en cours d'exécution il y aura alors réquisition du processeur en faveur du processus arrivant.

- **Priorité avec âge et quantum de temps :** Le principe de cet ordonnancement est le suivant, les processus sont élus en fonction de leur priorité. A chaque fin de « quantum », les processus en attente ont leur priorité qui se voit augmentée. Et une fois que l'un d'entre eux a une priorité plus élevé que le processus en cours alors il prend sa place d'exécution. Et une fois son « quantum » terminé, il reprend sa priorité initiale.



Priorité avec âge et quantum de temps (Source cours de M. Vantroys)

II.10/ Recherches pour la gestion des ressources

J'ai ensuite réalisé une phase de recherche concernant la gestion des ressources, dans mon cas les processus peuvent demander autant de ressources souhaitées. Nous avons alors parfois des interblocages car les processus obtiennent des accès exclusifs aux différentes ressources. Nous sommes donc en interblocage si chaque processus attend la libération d'une ressource qui est allouée à un autre processus de l'ensemble. Comme tous les processus sont en attentes, aucun ne pourra s'exécuter et donc libérer les ressources demandées par les autres. Ils attendront tous indéfiniment.

J'ai donc commencé à étudier différents algorithmes me permettant de mieux comprendre le problème de la gestion des ressources :

- **Le dîner des philosophes :** Nous avons par exemple 5 philosophes qui se retrouvent autour d'un table, chacun a une assiette devant lui et à gauche de chaque assiette il y a une fourchette. Chaque philosophe a trois états « penser », « être affamé » et « manger ». Lorsqu'un des philosophes a faim, il se met en état « affamé » et attend que les fourchettes soient libres car il doit utiliser la fourchette à sa droite et celle à sa gauche. Et si au moins une des fourchettes n'est pas libres alors il reste en état « affamé » avant de pouvoir retenter sa chance après un temps déterminé. Le problème a donc pour but de

trouver un ordonnancement pour que ces philosophes puissent manger chacun leur tour.

- **La politique de l'autruche :** cela consiste à éviter le problème en disant qu'il n'y a pas d'interblocage.

- **L'algorithme du banquier :** Cet algorithme est un algorithme d'évitement des interblocages qui s'applique dans le cas général où chaque type de ressources possède plusieurs instances. Lorsqu'un nouveau processus entre dans le système, il doit déclarer les ressources dont il aura besoin. Un processus n'a pas le droit d'utiliser toutes les ressources existantes. Lors de son exécution, quand un processus demande un ensemble de ressources, il y a ensuite une vérification pour s'assurer que le système ne sera pas grandement affecté.

II.11/ Variante pour la gestion des ressources

Il existe quatre conditions pour avoir un interblocage :

- Condition d'exclusion mutuelle : chaque ressource est soit attribuée à un seul processus, soit disponibles.

- Condition de détention et d'attente : les processus ayant déjà obtenu des ressources peuvent en demander de nouvelles.

- Pas de réquisition : les ressources déjà détenues ne peuvent être retirées de force à un processus. Elles doivent être explicitement libérées par le processus qui le détient.

- Condition d'attente circulaire : il doit y avoir un cycle d'au moins deux processus, chacun attendant une ressource détenue par un autre processus du cycle.

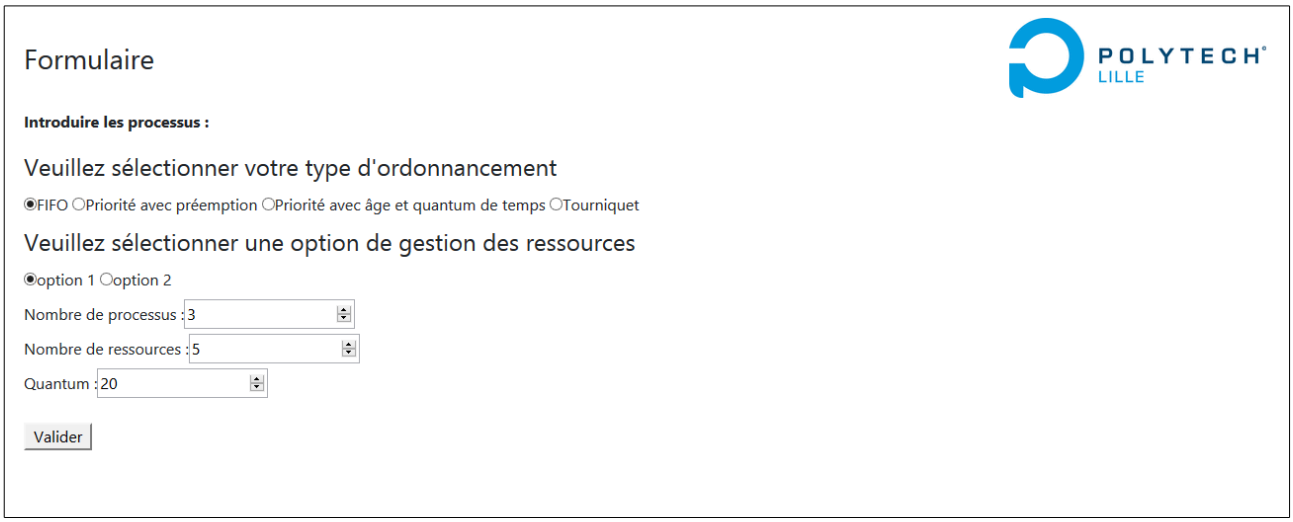
Après différentes recherches j'ai choisi de réaliser une variante ne permettant pas à un processus de demander plusieurs ressources à la fois. Et il a fallu regarder si chaque ressources demandées étaient bien relâchées par chaque processus. Si ce n'était pas le cas j'ai décidé de les relâcher systématiquement un peu avant la fin de leur temps d'exécution.

Une fois cette variante implémentée, j'ai ajouté dans le formulaire un groupe de boutons radio permettant de choisir son type de gestion des ressources. L'option 1 qui suit la politique de l'autruche et la 2 qui est la variante que je viens de présenter.

III/ TESTS DE L'APPLICATION WEB

Dans cette partie, je vais réaliser une démonstration en mettant quelques captures d'écrans de mon application Web. Je vais choisir le scénario suivant :

- Nous choisissons 3 processus
- Nous choisissons 5 ressources (même si nous ne les utilisons pas toutes lors de cette exemple)
- Puis nous choisissons par exemple l'ordonnancement de type FIFO
- Et mettons l'option 1 de gestion des ressources
- Le quantum ne sert à rien dans ce cas car nous prenons le FIFO
- Ensuite nous validons notre choix



Nous pouvons donc observer qu'il n'y a plus la première partie du formulaire comme expliqué précédemment. Une fois cela terminé, nous entrons les données suivantes :

Pour le premier processus :

- PID = 1
- Date de création = 0
- Priorité = 20
- Événements → Demande R1 à 10, Relâche R1 à 20 et on stoppe à 25. Ici on demande

alors 3 événements.

Ensuite pour le second processus :

- PID = 2
- Date de création = 5
- Priorité = 30
- Événements → Demande R2 à 5, Relâche R2 à 25 et on stoppe à 50. Dans ce cas nous demandons aussi 3 événements.

Et pour le dernier processus nous entrons cela :

- PID = 3
- Date de création = 10
- Priorité = 1
- Événements → Demande R4 à 10, Relâche R4 à 25, Demande R5 à 35, Relâche R5 à 55, et on stoppe à 100. On demande alors 5 événements.

Voici une visualisation de la seconde partie du formulaire.

P1 :

PID : 1

Date de creation : 0

Priorite : 20

Nombre d événements :

Evennement1 : Temps : 10 Operations : OP_CLAIM Semaphores : SEM_R1

Evennement2 : Temps : 20 Operations : OP_RELEASE Semaphores : SEM_R1

Evennement3 : Temps : 25 Operations : OP_STOP Semaphores : Choisir

P2 :

PID : 2

Date de creation : 5

Priorite : 30

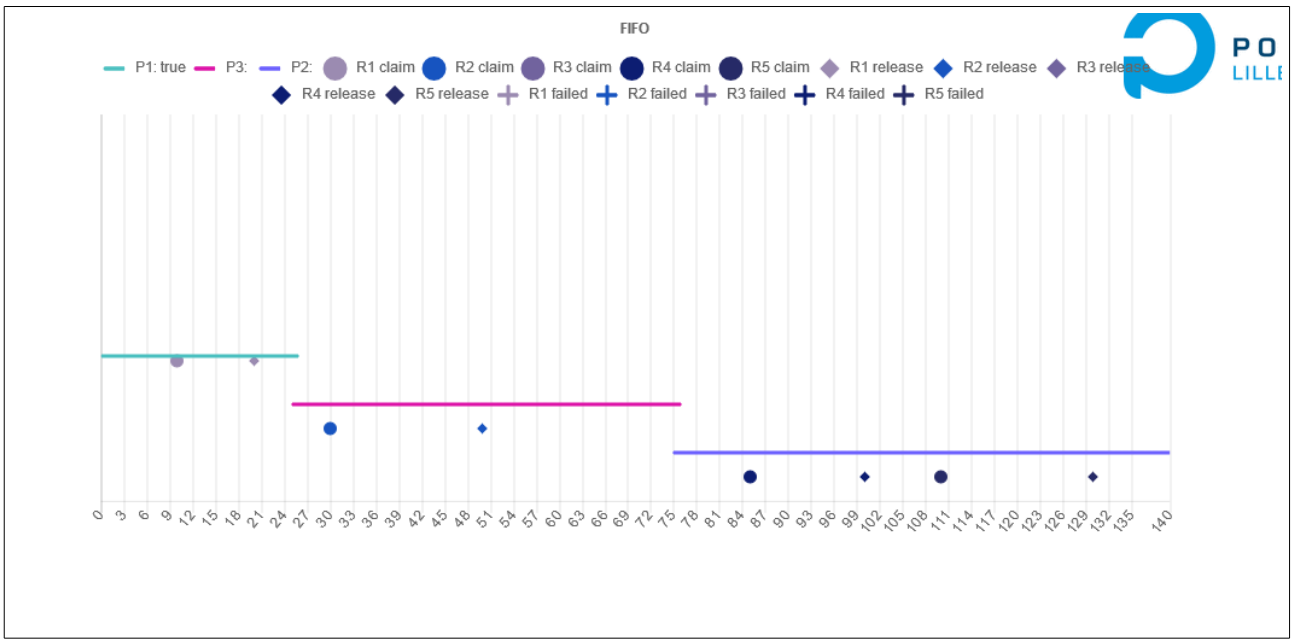
Nombre d événements :

Evennement1 : Temps : 5 Operations : OP_CLAIM Semaphores : SEM_R2

Evennement2 : Temps : 25 Operations : OP_RELEASE Semaphores : SEM_R2

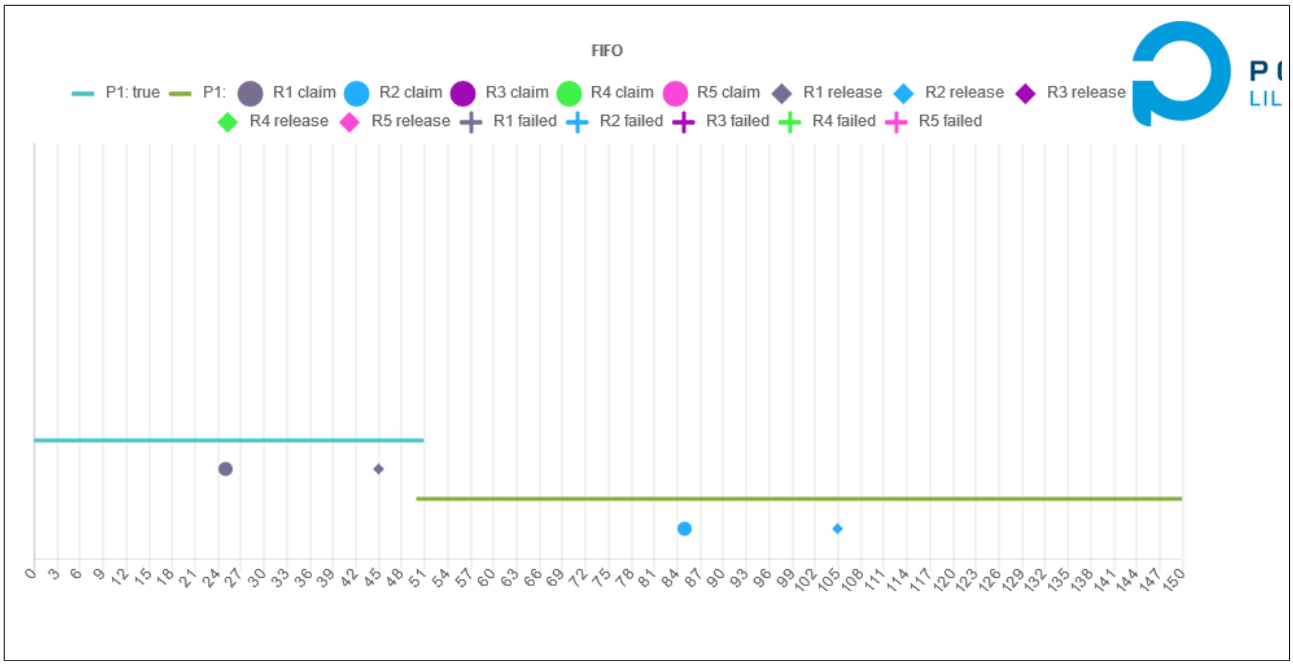
Evennement3 : Temps : 50 Operations : OP_STOP Semaphores : Choisir

Une fois toutes ces données entrées, nous validons et obtenons la capture d'écran suivante :



Comme nous pouvons le voir, il y a une légende et par exemple chaque processus possède sa couleur, et chaque ressource à aussi sa propre couleur. De plus, nous pouvons observer que la simulation fonctionne correctement car c'est bien le deuxième processus qui commence, puis le premier et enfin le dernier. Et si nous regardons les différentes dates de création nous pouvons observer que cela est valide.

Ensuite, j'ai entré des valeurs avec l'option 2 pour la gestion des ressources et j'avais mis que le premier processus demander une ressource et ne la relâchait jamais. Ainsi, avec l'option 2 on a quand même un relâchement qui se fait automatiquement.



IV/ DIFFICULTÉS RENCONTRÉES ET LIMITES DU PROJET

Au cours de ce projet, j'ai rencontré plusieurs problèmes tels que pour dynamiser le formulaire par un script JavaScript, il faut savoir manipuler correctement le « DOM » du fichier HTML. Le DOM permet à des scripts d'examiner et de modifier le contenu du navigateur web. Ce fut donc compliqué car il est plus simple de gérer des éléments statiques que de gérer des éléments ajoutés dynamiquement. De plus, j'ai eu quelques difficultés pour gérer les événements dans la partie de l'interface processus.

Au départ je ne savais pas par où commencer sachant que je ne connaissais pas le JavaScript. Je me suis égaré à de nombreuses reprises en essayant de coder en PHP par exemple au lieu de coder en JavaScript. Je ne connaissais pas les différentes méthodes pour enregistrer les données du formulaire par exemple.

Concernant la recherche de bibliothèque pour afficher le chronogramme j'ai pris du temps à en trouver une permettant de réaliser un traçage correct des chronogrammes.

Globalement, mon projet répond au cahier des charges mais il y a des points d'améliorations possibles :

- Essayer de mettre en place un historique des chronogrammes

- Mettre un bouton permettant d'ajouter un chronogramme juste en dessous de celui venant de s'afficher

- Lorsque l'on passe la souris sur les différents type d'ordonnancement, nous pourrions avoir une définition qui s'affiche

CONCLUSION

Pour finir, ce projet a donc abouti à une application Web exploitable, mais perfectible. J'ai pu développer des compétences techniques. Les difficultés que j'ai rencontré sont avant tout techniques puisqu'il m'a fallu apprendre de nouveaux langages de programmation : JavaScript et CSS. Je tiens donc à ajouter qu'une grosse partie de mon travail a donc consisté à réaliser une phase de recherche d'informations concernant ces langages, mais aussi concernant la gestion des ressources.

J'ai donc beaucoup appris et si un projet de même envergure était à refaire je pourrais m'y prendre d'une meilleure manière. Ce projet m'a permis de réaliser une remise en question concernant mes différents chois techniques. Les phases de recherches d'erreurs et de tests ont duré énormément de temps. A l'avenir je saurais organiser mon temps d'une meilleure manière.

BIBLIOGRAPHIE

- « Systèmes d'exploitation » d'Andrew Tanenbaum
- « Solutions temps réel sous Linux » de Christophe Blaess
- Cours de systèmes de M. Vantroys
- https://www.w3schools.com
- https://3wa.fr
- https://developer.mozilla.org