Projet SY40 Automne 2020

PROBLEMATIQUE

Dans une confiturerie, la gestion de la chaîne pour le remplissage des bocaux est toujours problématique. On doit bien organiser les processus des bocaux et valves afin d'effectuer efficacement le remplissage.

Professeur : Philippe DESCAMPS
Réalisé par : Yuan Cao

Sommaire

INTRODUCTION

Dans une confiturerie, la gestion de la chaîne pour le remplissage des bocaux est toujours problématique. On doit bien organiser les processus des bocaux et valves afin d'effectuer efficacement le remplissage. Dans le cours SY40, on a appris les outils nécessaires pour faire un projet qui vise à le réaliser. Le projet nous demande d'utiliser deux moyens diffiérents : des processus et des sémaphores, des threads et des moniteurs pour réaliser le remplissage de deux types des bocaux.

1.Présentation du projet

1.1 Description

Le projet se divise en trois parts :

1. Une confiturerie dispose d'une chaîne pour le remplissage des bocaux. Un processus **Bocal** assure l'arrivée des bocaux. Un autre processus **Valve** fait ouvrir la valve et utilise un processus **Horloge** pour mesurer un délai, le délai est décidé par l'utilisateur. Après le délai il ferme la valve et alors signale au processus Bocal que le remplissage est terminé, et qu'il peut enlever le bocal et en placer un nouveau.

   On utilise des processus et des sémaphores IPC pour le réaliser.

2. On ajoute une autre processus **Bocal_2** et assure qu'un seul processus à la fois, Bocal ou Bocal_2, place un bocal dans l'unité de remplissage. De plus, les nouveaux bocaux BocalGrand (processus Bocal_2) sont deux fois plus grands que les anciens BocalGrand. Le processus Valve doit donc obtenir un délai deux fois plus long.

   On utilise des processus et des sémaphores IPC pour le réaliser.

3. Une confiturerie dispose d'une chaîne pour le remplissage des bocaux. Un thread **Bocal** assure l'arrivée des bocaux. Un autre thread **Valve** fait ouvrir la valve et utilise un thread **Horloge** pour mesurer un délai, le délai est décidé par l'utilisateur. Après le délai il ferme la valve et alors signale au thread Bocal que le remplissage est terminé, et qu'il peut enlever le bocal et en placer un nouveau.

   On ajoute un autre thread **Bocal_2** et assure qu'un seul thread à la fois, Bocal ou Bocal_2, place un bocal dans l'unité de remplissage. De plus, les nouveaux bocaux BocalGrand (thread Bocal_2) sont deux fois plus grands que les anciens BocalGrand. Le thread Valve doit donc obtenir un délai deux fois plus long.

   On utilise des threads et des moniteurs pour le réaliser.

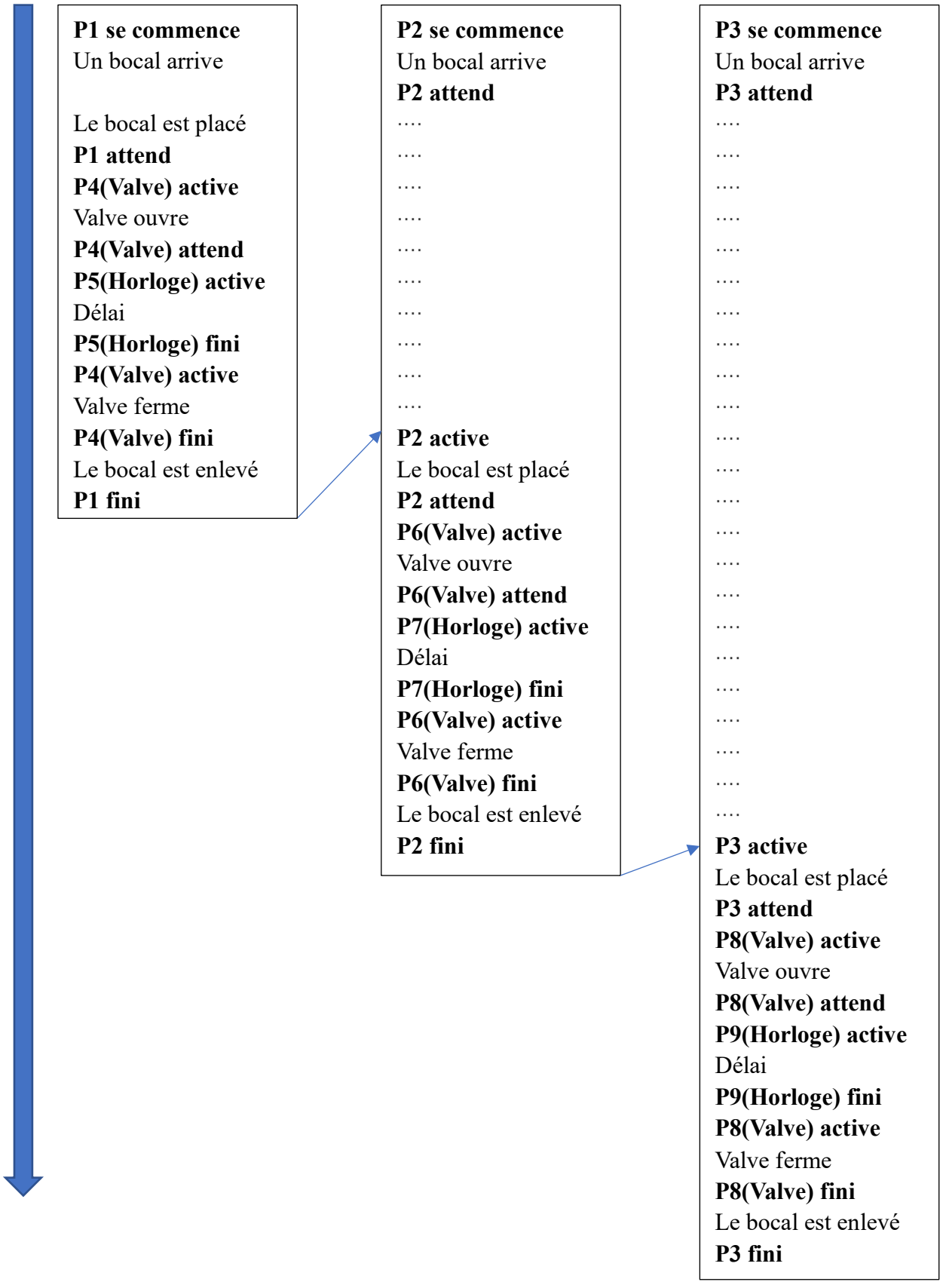
1.2 Problématique

Le problématique est d'assurer une bonne synchronisation entre les processus bocal(bocal_2), valve et horloge pour effectuer le remplissage efficace des plusieurs bocaux.

2. Conception

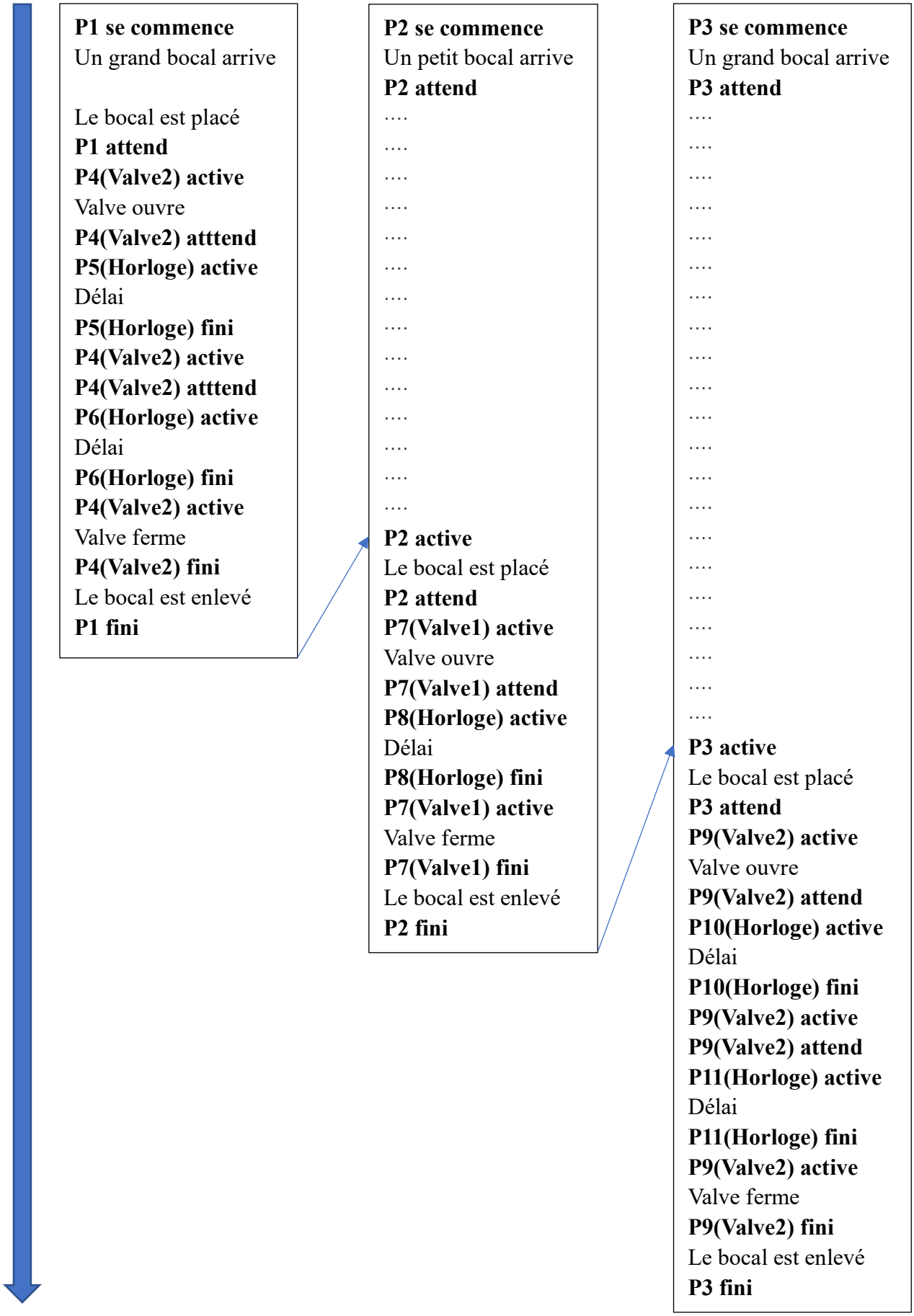
1. Pour la partie 1, on suppose qu'il y a trois bocaux à remplir.

   Le placement des bocaux sont **aléatoires**.



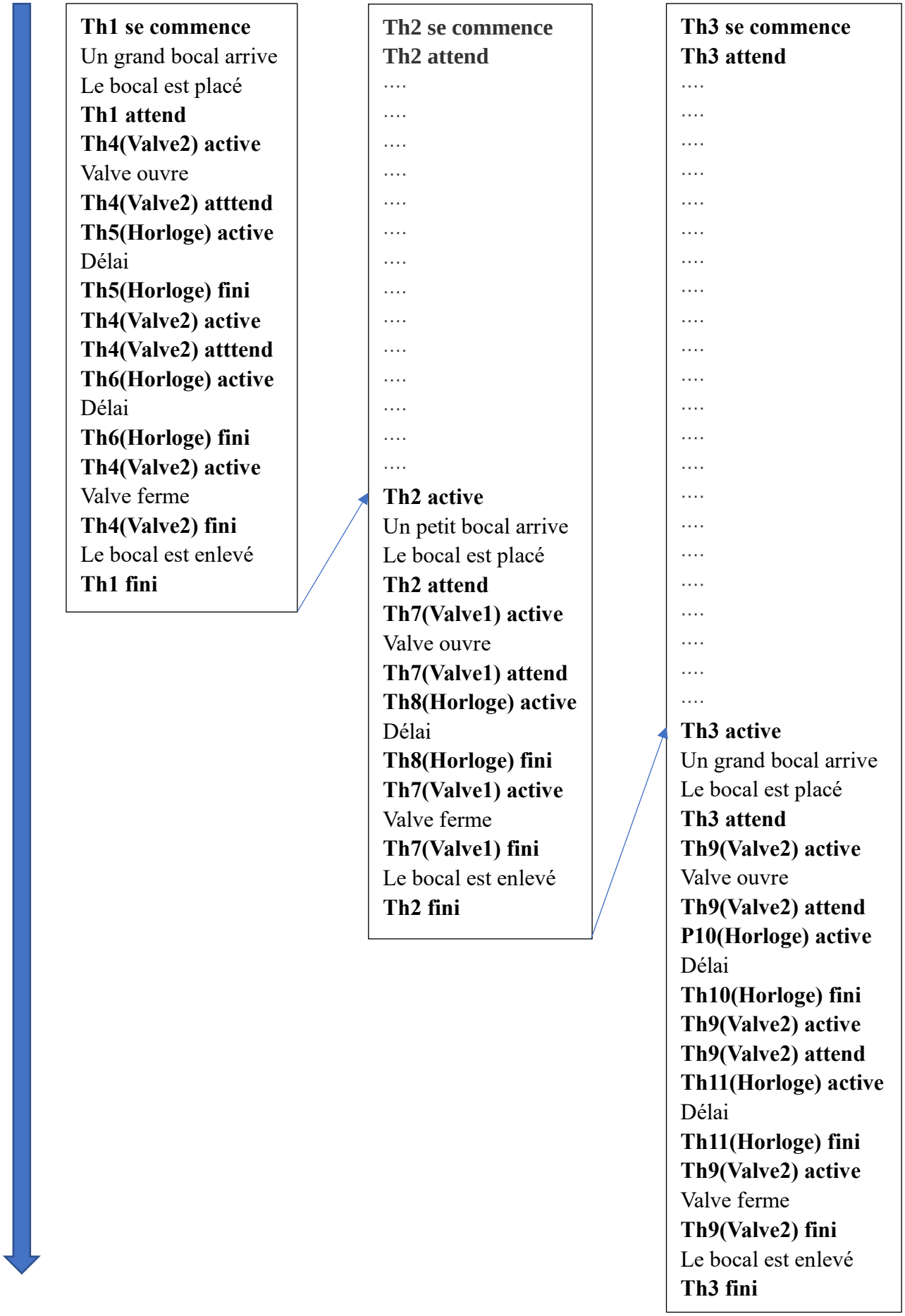
2.Pour la partie 2, on suppose qu'il y a un bocal petit et deux bocaux grands à remplir.

Le placement des bocaux n'importe quelles tailles sont **aléatoires**.



3.Pour la partie 3, on suppose qu'il y a un bocal petit et deux bocaux grands à remplir.

Les arrivées des bocaux n'importe quelles tailles sont **en ordre**.



3. Implémentation

3.1 Outils

1. Le langage C : J'ai utilisé le langage C pour réaliser ce projet. En outre, j'ai utilisé ce que j'ai appris dans le cours : des threads et des moniteurs, des processus et des sémaphores IPC.

2. L'environnement de développement : Mon projet a été fait dans un environnement macOS. Je me suis servi du compilateur "gcc".

3.2 Structure

con1.c

Il vise à resoudre le problème a) en utilisant des processus et des sémaphores IPC.

con2.c

Il vise à resoudre le problème b) en utilisant des processus et des sémaphores IPC.

con3.c

Il vise à resoudre le problème c) en utilisant des threads et des moniteurs.

Makefile

Il vise à compiler les programmes en entrant la commande "make".

Le but de ce projet est de faire se procéder tous les processus dans l'ordre, donc j'ai utilisé les outils des synchronizations.

Les sémaphores :

Dans con1.c, j'ai utilisé 5 sémaphores :

(0) Il est initialisé à 1. Ce sémaphore assure qu'une fois il y un seul bocal qui est placé.

(1) Il est initialisé à 0. Ce sémaphore est mis au début du processus **valve_oper.** Donc le **valve_oper** dort au début jusque le bocal le réveille pour le remplissage. Il assure qu'une fois un seul bocal peut être mis dans le processus **valve_oper** et aussi la valve ne doit être ouverte qu'après le placement d'un bocal.

(2) Il est initialisé à 0. Ce sémaphore est mis au début du processus **horloge.** Donc le **horloge** dort au début jusque le bocal qui est dans le processus **valve_oper** le réveille pour le mesurer un délais qui est entré par l'utilisateur au début. Il assure aussi qu'une fois un seul processus **horloge** se déroule comme le remplissage.

(3) Il est initialisé à 0. Ce sémaphore est mis dans le processus **valve_oper** après l'ouverture de la valve et avant la fermeture. Donc le processus **valve_oper** dort entre eux. Il assure aussi que la valve ne doit être fermée qu'après le remplissage.

(4) Il est initialisé à 0. Ce sémaphore est mis dans le processus **bocal** avant l'enlèvement du bocal. Donc le processus **bocal** dort après le placement. Il ne doit être réveillé par le processus **valve_oper** qu'après le remplissage est fini et la valve est fermée.

Dans con2.c, j'ai utilisé 6 sémaphores :

(0) Il est initialisé à 1. Ce sémaphore assure qu'une fois il y un seul bocal n'importe quelle taille qui est placé. (Le même que con1.c)

(1) Il est initialisé à 0. Ce sémaphore est mis au début du processus **valve_oper01.** Donc le **valve_oper01** dort au début jusque le petit bocal le réveille pour le remplissage. Il assure qu'une fois un seul petit bocal peut être mis dans le processus **valve_oper01** et aussi la valve ne doit être ouverte qu'après le placement d'un bocal.

(2) Il est initialisé à 0. Ce sémaphore est mis au début du processus **horloge.** Donc le **horloge** dort au début jusque le bocal qui est dans le processus **valve_oper** le réveille pour le mesurer un délais qui est entré par l'utilisateur au début. Il assure aussi qu'une fois un seul processus **horloge** se déroule comme le remplissage. (Le même que con1.c)

(3) Il est initialisé à 0. Ce sémaphore est mis dans le processus **valve_oper** après l'ouverture de la valve et avant la fermeture. Donc le processus **valve_oper** dort entre eux. Il assure que la valve ne doit être fermée qu'après le remplissage. (Le même que con1.c)

(4) Il est initialisé à 0. Ce sémaphore est mis au début du processus **valve_oper02.** Donc le **valve_oper02** dort au début jusque le grand bocal le réveille pour le remplissage. Il assure qu'une fois un seul grand bocal peut être mis dans le processus **valve_oper02** et aussi la valve ne doit être ouverte qu'après le placement d'un bocal.

(5) Il est initialisé à 0. Ce sémaphore est mis dans le processus **bocal** avant l'enlèvement du bocal. Donc le processus **bocal** dort après le placement. Il ne doit être réveillé par le processus **valve_oper** qu'après le remplissage est fini et la valve est fermée.

Les mutex :

Dans con3.c :

Il y a les threads pour les bocaux petits et grands. On crée les threads **oper_valve** pour les petits bocaux et **oper_valve_2** pour les grands bocaux qui ont deux itérations du thread **horloge**. Il y a aussi les threads **horloge** pour mesurer un délai qui est entré par l'utilisateur au début.

J'ai utilisé 2 mutex et 5 variables de condition :

(mutex_forall) Il assure que les threads va s'executer successivement et une fois un seul thread peut se procéder.

(mutex) Il est utilisé du début à la fin et assure qu'une fois il y a un seul thread qui se procède.

(valve) Cette variable de condition est utilisée comme le sémaphore(1) dans con2.c, qui bloque le thread au début qui utilise la fonction **oper_valve** et attend que le petit bocal est placé et le réveille.

(valve_2) Cette variable de condition est utilisée comme le sémaphore (4) dans con2.c, qui bloque le thread au début qui utilise la fonction **oper_valve_2** et attend que le grand bocal est placé et le réveille.

(horloge) Cette variable de condition est utilisée comme le sémaphore (2) dans con2.c, qui bloque le thread au début qui utilise la fonction **oper_horloge** et attend que la valve est ouverte et le réveille pour mesurer un délai.

(fermer) Cette variable de condition est utilisée comme le sémaphore (3) dans con2.c, qui bloque le thread jusque le délai est fini et après ferme la valve.

(enleve) Cette variable de condition est utilisée comme le sémaphore (5) dans con2.c, elle bloque le thread avant l'enlèvement du bocal. Il ne doit être réveillé qu'après le remplissage est fini et la valve est fermée.

Programme principal (main)

Au début, le programme nous demande d'entrer les nombres des bocaux ou des petits bocaux et des grands bocaux. On doit après entrer le temps du remplissage pour commencer.

Le programme va afficher les informations s'il y a une erreur dans la création et initialisation des sémaphores, threads etc.

En fin, on attend que tous les processus ou threads se terminent et après on fait la destruction des sémaphores et libère toutes les ressources.

4. Conclusion

Ce projet me permet de mettre en pratique les connaissances acquises en cours, TD, TP de l'UV SY40. Grâce à celui, je suis sûr que j'ai plus de confiance pour résoudre les problèmes quotidiens, surtout les problèmes qui ont plusieurs tâches à faire et donc ont besoin de la synchronisation.