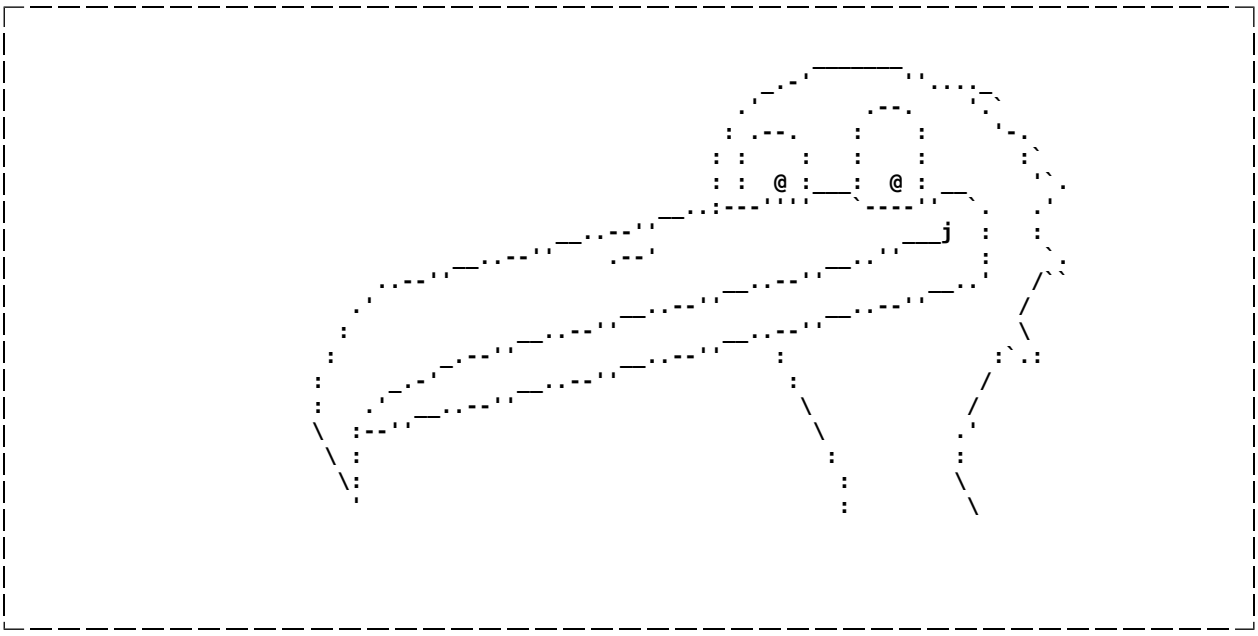
PéliKanbon

Projet de LO41

Bretegnier Paul-Émile
Lamielle Antoine

Semestre d'Automne 2014



Introduction

Dans le cadre de l'UV LO41, nous avons dû développé un programme capable de simuler le principe de la méthode KANBAN. Cette méthode est mise en place entre deux postes de travail et limite la production du poste en amont aux besoins exacts du poste aval. En d'autres termes, la production des pièces est dictée par l'aval et non par l'amont. Ainsi, cette technique permet de limiter au maximum les en-cours durant les processus de fabrication.

À travers ce rapport, nous expliquerons la démarche que nous avons suivie, les méthodes que nous avons choisis mais aussi les problèmes que nous avons pus rencontrer.

Sommaire

Introduction....2
1.Démarche....3
2.Fonctionnement général....3
3.Initialiser le programme....3
4.Fonctionnement de la chaîne de production....5
5.Fonctionnement d'un atelier....6
6.Fermeture du programme....7
7.Difficultés rencontrées....7
8.Améliorations possibles....7

1. Démarche

Afin de réaliser ce projet, nous avons commencé par réfléchir sur quels outils nous allions utilisé pour modéliser notre programme. Nous avons pris en compte les avantages et inconvénients de plusieurs méthodes en terme de rapidité et facilité d'exécution, mais aussi en s'arrangeant au mieux pour optimiser les ressources utilisées par le programme.

Comment est-ce que la chaîne de production va-t-elle être générée ? Comment gérer les liens entre chaque poste de travail ? Comment vont-ils communiquer ?

Nous avons donc décidé d'utiliser une combinaison d'un objet IPC (pour la communication) avec un ensemble de processus (rprésentant les ateliers de travail) généré à l'aide de *fork*, et rassemblé sous la forme d'un arbre.

2. Fonctionnement général

Le programme démarre avec l'exécution du *padre*, qui chapeaute l'ensemble des actions du programme. Celui-ci va se charger de générer l'objet IPC qui permettra de faire communiquer les différents postes de travail. Puis grâce à l'IHM intégrée, l'utilisateur peut définir une nouvelle chaîne de production donc un nouvel arbre, en choisissant le nombre de « niveau » ou de profondeur ainsi que le nombre de « nœud fils » pour chaque nœud (poste de travail).

Ce choix permet de générer un arbre avec un taille variable sans trop de complexité, sans pour autant permettre de définir un nombre de « nœuds fils » pour chaque nœud.

L'utilisateur peut ensuite :

- Définir le nombre de produit dont il a besoin sur cette chaîne de production
- Afficher l'arbre des postes de travail
- Détruire la chaîne (et quitter le programme)

3. Initialiser le programme

Le *padre* génère un objet IPC (équivalent au tableau de lancement) qui permettra aux différents postes de travail de communiquer. En effet, un poste voulant envoyer une requête, un conteneur, ou toute autre information devra envoyer un message ipc, contenant l'identifiant du destinataire, le type de requête, et le message.

Un arbre est généré avant les *forks* du *padre*. Chaque nœud de cet arbre pointe vers son parent (client) et ses enfants (fournisseurs). De plus, chacun contient un identifiant unique d'atelier ainsi que les *pids* de ses clients et de son parent.

Ensuite les *forks* du *padre* ont lieu afin de créer l'ensemble des ateliers :

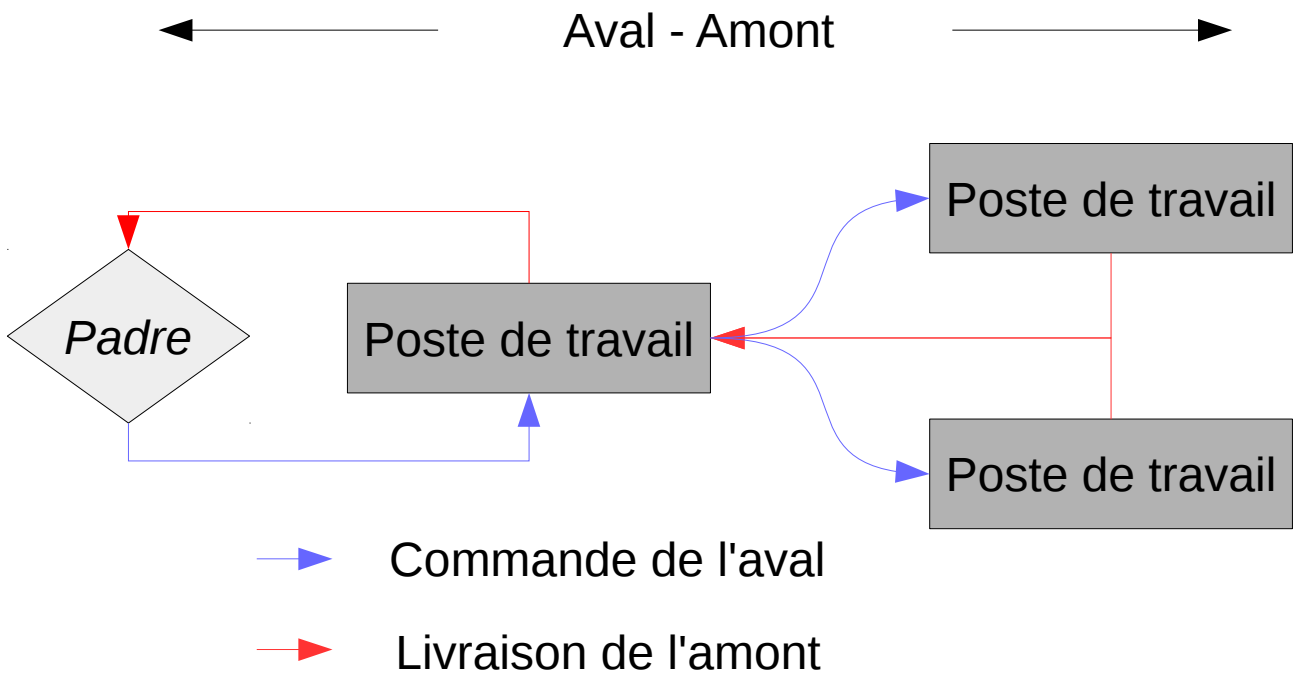
- Les postes de travail se génèrent et s'initialisent localement en fonction de leurs paramètres ;
- Les postes en amont communiquent le nombre de pièces qu'ils produisent par conteneur à leur client en aval ;
- Une fois près, ils en informent le *padre* (envoie d'un message *Ready*) ; lorsque celui-ci a reçu confirmation de tous les ateliers, il leur envoie un message *Start*, leur permettant d'entrer dans la boucle de production.

4. Fonctionnement de la chaîne de production

De base, chaque poste de travail (PdT) dispose de deux conteneurs pleins de chacun de leur fournisseur. Lorsque qu'un PdT ouvre un conteneur, il envoie immédiatement une commande de conteneur au fournisseur concerné, afin de n'être jamais en manque de pièce.

Quand l'utilisateur envoie une requête de fabrication pour un produit au *padre*, l'ordre remonte au poste de travail racine. Celui-ci va commencer à fabriquer ses pièces puis en fonction des règles de fonctionnement ci-dessus, va envoyé une requête à ses fournisseurs pour avoir toujours 2 conteneurs pleins. Ceux-ci font de même avec leurs propres fournisseurs et ainsi de suite de manière récursive.

Illustration 1: la communication dans la chaîne de production



5. Fonctionnement d'un atelier

Le fonctionnement d'un atelier se répartit en deux étapes principales :

- l'initialisation (comme expliquée dans l'initialisation du programme) ;
- la boucle principale, que l'on va voir à présent.

Une fois démarré, cette boucle principale continuera de tourner tant que l'atelier ne reçoit pas de message d'arrêt de la part du *padre*.

S'il doit produire un conteneur, le poste de travail vérifie qu'il possède bien au moins une pièce de chacun de ses fournisseurs, le cas échéant il attend d'en avoir (en pratique cela ne s'est, et ne doit jamais se produire).

Puis, il commence à travailler en prenant un pièce dans le conteneur ouvert de chacun de ses fournisseurs. S'il est vide, l'atelier ouvre un nouveau conteneur et envoie immédiatement une commande pour un nouveau conteneur au fournisseur concerné. Il crée alors un nouveau produit. Si le nombre de produits créés est égale au nombre de pièces que l'atelier peut mettre dans un conteneur il en charge un avec ses pièces et l'envoie à son client.

Enfin, l'atelier gère l'ensemble des messages qu'il reçoit. Soit quelque chose est à produire, alors on relève les messages sans ce mettre en pause ; Soit rien est à produire, et le poste de travail patiente jusqu'à ce qu'un message soit reçu. Lors de la réception d'un message, l'atelier analyse le type de message et le traite en conséquence.

Le traitement des message reçu par un PdT s'effectue à la fin de la boucle principale pour éviter de repasser par la production (et potentiellement générer de nouvelles requêtes) une fois que la séquence d'arrêt est lancée.

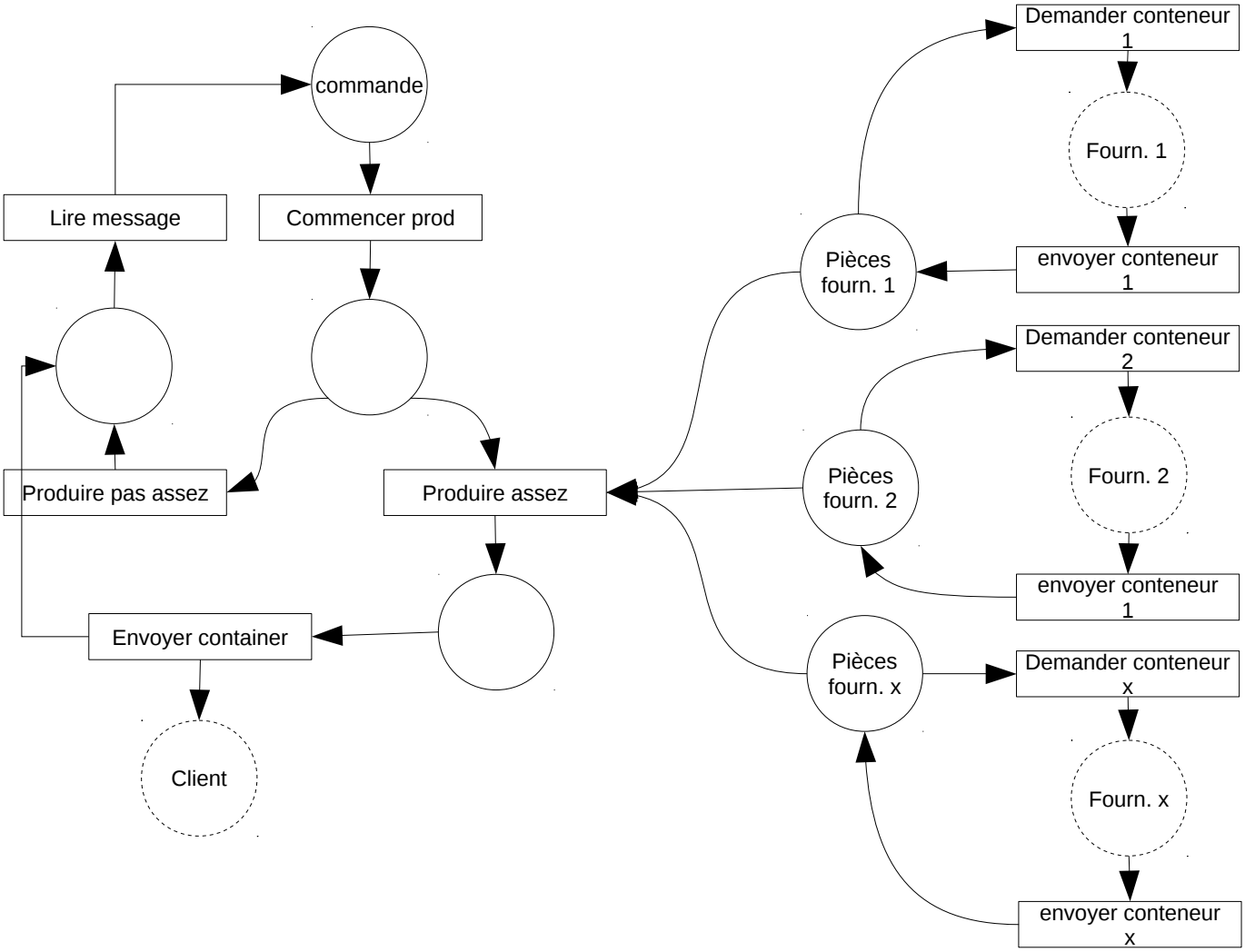


Illustration 2: Diagramme de pétri d'un poste de travail

6. Fermeture du programme

Quand l'utilisateur désire fermer le programme, il doit d'abord « détruire » la chaîne de production. Le *padre* envoie un message de fermeture à tous les postes de travail et attend leur confirmation pour détruit l'arbre. L'objet IPC est détruit en dernier, après la fermeture de l'IHM.

7. Difficultés rencontrées

Le plus délicat fut de définir l'initialisation du programme, ou comment créer l'ensemble de la chaîne. Ainsi que la correction de bogues concernant la synchronisation lié à la communication inter-processus.

8. Améliorations possibles

On pourrait imaginer un affichage graphique élaboré permettant de consulter en temps réel l'état de la chaîne. De plus on pourrait modifier la génération de l'arbre afin de définir manuellement, ou à l'aide d'un fichier un nombre personnalisé de client pour chaque nœud.