Mini Projet SYSTEME L3 Classique

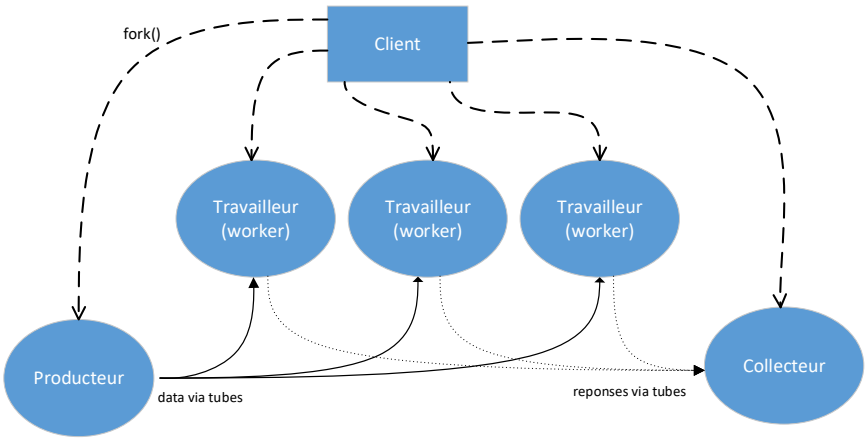
Mars 2020, LEGOND Fabrice

I. Présentation d'un Modèle de « word count » parallèle

On souhaite implémenter à l'aide de tubes et de processus multiples une sorte de groupe (Pool) de travailleurs (Worker) destiné à exécuter des traitements parallèles de comptage (identique à la commande « `wc` »). Les acteurs de ce « `wc` » parallèle seront des processus Unix.

Le client est un processus à part qui va créer tous les processus et attendra la fin de tous ses enfants. Ce sera le premier né de la solution et le dernier à mourir. C'est celui qui sera exécuté en tapant le nom dans le bash. Une solution plus optimisée consisterait à faire aussi travailler le processus client mais nous ne le ferons pas.

Chaque processus enfant « `travailleur` » pourra exécuter le traitement des données sous formes de blocs successifs (sans mourir) qu'il recevra du « `producteur` » et renverront les résultats du comptage au « `collecteur` ».



Les liens/dialogues entre le `Producteur`, les `Travailleurs` et le `Collecteur` se feront de préférences à base de tubes. Il peut être envisageable que la communication se fasse par d'autres moyens qui devront permettre l'échange de données. Le `client` enverra une commande démarrage au `producteur` et le résultat sera affiché `collecteur`.

Vous ferrez particulièrement attention à la discussion `producteur` / `travailleurs` / `collecteur` qui pourra, selon votre architecture, requérir peut-être plusieurs canaux. Rappelons que ces processus ne sont pas « père/fils ».

Il vous appartient de définir le protocole pour pouvoir envoyer ce type de message et obtenir une réponse sous forme de texte. Le protocole inclus les contenus et les formats des messages qui circuleront dans les différents types de tubes.

La commande « wcp » aura les paramètres suivants :

```
wcp type_comptage nb_enfants nb_lg liste_fichiers
```

où les paramètres sont :

- type_comptage : ce qu'il faut compter dans les fichiers textes
  - « c » : les caractères, chaque caractère(insensibilité à la casse) / chiffre différent devra être compté ;
  - « w » : les mots, chaque mot différents (insensibilité à la casse) devra être compté ;
  - « s » : les séparateurs, chaque séparateur différent devra être compté (\n, \t, espace, ',', ';', '!',…) ;
- nb_enfants : le nombre de processus « travailleurs » à créer. Si 0, le « client » devra créer un nombre de « travailleurs » égale au nombre de cpu – 2 (minimum 2). Le nombre maximum de « travailleurs » (quelle que soit la situation) sera 8.
- nb_lg : taille des blocs en nombre de lignes
- liste_fichiers : liste de tous les fichiers à lire

Notes importantes :

- les fichiers seront des fichiers textes sans accents (pour faciliter la gestion de l'insensibilité à la casse) ;
- chaque ligne ne dépassera pas 1024 caractères.
- chaque mot n'aura pas plus de 50 caractères ;
- vous ne pourrez compter au-delà de 2^{32} soit ~4 milliards de chaque éléments
- vous pourrez ou non gérer l'erreur que pourrait générer ce dépassement de capacité (**bonus**) ;
- vous veillerez à bien décrire les contraintes que vous mettez sur l'écriture et la lecture des tubes ;
- vous veillerez à bien décrire le format des différents messages ;
- vous indiquerez si vous utilisez des tubes anonymes ou des tubes nommés et combien et dans quel sens. Cela dépendra de votre solution ;
- vous indiquerez comment se termine les processus et dans quel ordre ;

Par exemple, si un fichier contient le ligne :

```
Nous sommes le 8 mars 2020 et il y a du vent.
```

L'option « c » donnera comme résultat :

N	2
O	2
U	2
S	4
M	3
E	4
…	
2	2

L'option « w » donnera comme résultat :

Nous	1
Sommes	1
Le	1
…	
2020	1

L'option « s » donnera comme résultat :

.	1

II. Questions concernant le Modèle du programme « wpc »

Le rapport contiendra au moins :

1) Représenter l'architecture globale du système pour 2 workers en faisant apparaître précisément les tubes utilisés et les sens de circulation des données sur ces tubes.
2) Définir la structure des messages (ie nombre de lignes, format de chaque ligne) dans chaque type de tube que vous avez défini. S'il s'agit simplement de données sans structure (raw), dites-le.
3) Donner les séquences échangées lors du traitement d'une commande.

Concernant le code :

4) Implémenter le code des différents acteurs en utilisant des tubes et l'utilisation de fork / exec / pipe / dup / ...
5) Tester votre application avec différents fichiers. Expliquer vos tests.

III. Documents et fichiers à remettre

Il y aura un rapport à remettre correspondant aux diagrammes et aux descriptions des choix techniques (tubes anonymes, tubes nommés, fork/exec, relation père/fils, …). Ce fichier sera un fichier pdf en plus du code.

Tout devra être remis sous forme d'un fichier zip contenant le fichier pdf et les sources C, ainsi qu'un fichier de script make.sh pour compiler votre code (avec un Makefile si vous savez le faire – ce qui apportera un petit bonus).

Structure de l'archive (numetudiant_nometudiant_prenometudiant.zip) :

- rapport.pdf
- make.sh
- fichier(s) source(s) c