IN329 : BE

SI

`http://personnel.supaero.fr/garion-christophe/IN329` À rendre pour le 18 janvier 2008

Ce document est le sujet de bureau d'études pour le module IN329 « Système d'exploitation ». Ce travail est individuel. Il propose de généraliser la notion de *pipe*.

---

1 Présentation du problème

Un *pipe* dans Unix est un moyen de communication unidirectionnel entre deux processus : un émetteur et un récepteur. Diverses extensions de cette notion ont été envisagées :

- T : unidirectionnel avec un émetteur et deux récepteurs ;
- *pipe* bidirectionnel entre deux processus ;
- ...

On se propose de généraliser le mécanisme de communication à plusieurs processus et plusieurs *pipes*, les processus pouvant communiquer deux à deux à l'aide de *pipes* unidirectionnels. L'ensemble des processus et des *pipes* forme un graphe orienté dont les sommets représentent les processus et les arcs (orientés) représentent les *pipes* unidirectionnels. Un exemple est représenté sur la figure 1 : les processus sont P_1, P_2, P_3 et P_4 et les *pipes* a, b, c et d.

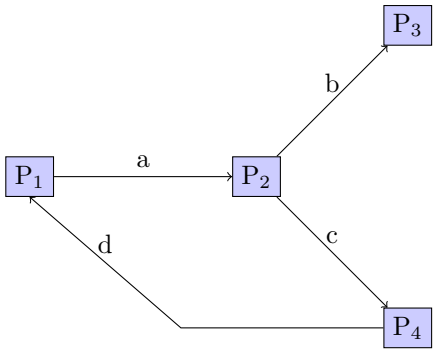

FIG. 1 – Un graphe orienté *pipes*/processus

2 Travail à effectuer

Vous devez écrire une application `multi_pipe` en C qui gère ce mécanisme de communication multi-pipes. Nous supposerons dans un premier temps pour simplifier le problème que les processus exécutent des programmes sans arguments et que l'ensemble « processus/*pipes* » est défini dans un fichier selon un format décrivant un processus par trois lignes :

- la première ligne donne le nom du programme exécuté par le processus,
- la deuxième ligne donne la liste des *pipes* d'entrée du processus,
- la troisième ligne donne la liste des *pipes* de sortie du processus.

L'application `multi_pipe` recevra comme argument le nom du fichier de description.

Par exemple, on pourra exécuter :

```
multi_pipe desc.txt
```

où le fichier desc.txt contiendra :

```
P1
in:d
out:a
P2
in:a
out:b,c
P3
in:b
out:
P4
in:c
out:d
```

On souhaite également que le programme `multi_pipe` puisse réaliser des traitements arithmétiques. Nous ne considérerons que le calcul $(x + y) \times (y - z)$. Le graphe associé seront donc celui représenté sur la figure 2

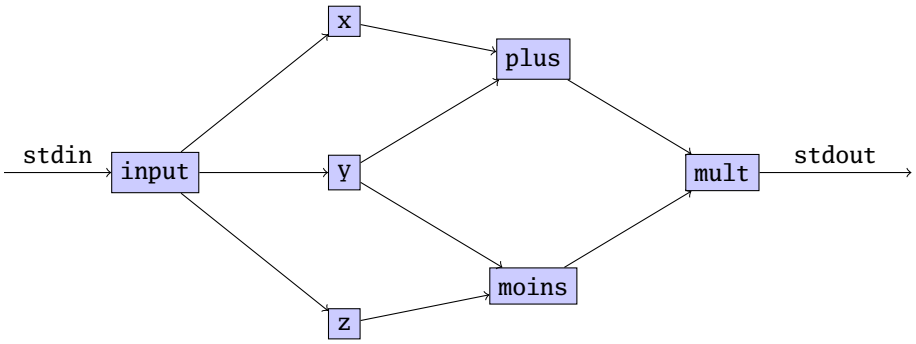

FIG. 2 – Graphe de l'opération $(x + y) \times (y - z)$

Il faudra donc écrire également les programmes `input`, `plus`, `moins`, `mult`, `x`, `y` et `z`. On supposera que les valeurs de x, y et z seront lues par un seul processus conversationnel pour éviter les conflits sur `stdin`.

Les éventuelles extensions du BE pourront être par exemple l'utilisation de programmes avec arguments, une extension des possibilités d'évaluation d'expressions arithmétiques etc.

Quelques remarques :

- on supposera que les commandes sont données avec leur *nom complet* comme par exemple `/bin/ls` ;
- on pourra trouver les pages de manuel des appels système Unix/Linux dans [1] ;
- pour le *parsing* des chaînes de caractères, on pourra utiliser la fonction `strtok` de la librairie `string`.

3 Documents à rendre

Vous devez rendre les documents suivants :

- une archive ZIP ou tar.gz comprenant vos sources C commentés et un fichier Makefile ou un fichier `README.txt` expliquant comment compiler votre projet et l'exécuter. La compilation de vos sources devra produire un exécutable utilisable sur les machines Linux du Centre Informatique ;
- quelques exemples d'exécution de votre programme ;

– un rapport succinct de 5 pages environ présentant l'architecture de votre application, les solutions proposées, les limitations de votre solution et l'éventuelle bibliographie utilisée. Ce rapport devra être au format PDF.

Ces documents sont à envoyer par mail à `garion@supaero.fr` pour le **18 janvier 2008**.

Références

[1] Online manual pages of Unix/Linux. `http://www.phpman.info/`.