TP Noté - ASR5 système d'exploitation

2h30 tout document autorisé

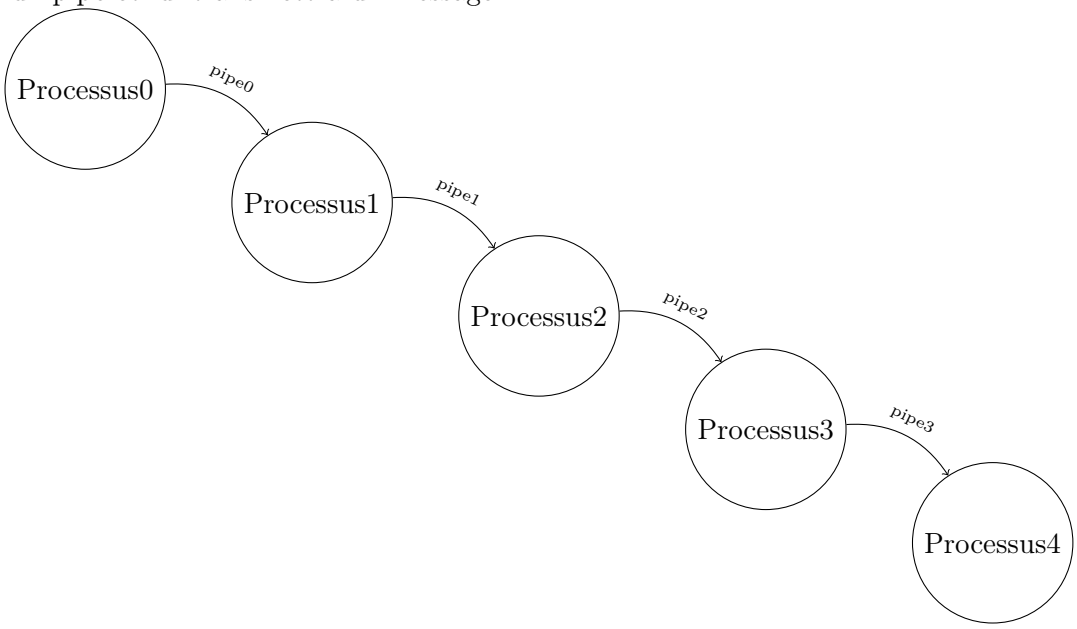
2 mars 2017

Les programmes qui vous sont demandés doivent être écrits en C ou C++ sur les machines de la salle. Vous n'êtes pas autorisés à échanger d'information (notamment par le réseau), ni à vous connecter aux réseaux extérieurs de la salle.

Les machines de la salle vous permettent de vous connecter localement avec le login moi et le mot de passe moi. Vous sauvegarderez votre travail sur ce compte, et vous le déposerez à la fin sous la forme d'une archive `nom_prenom.tar.gz` sur le serveur `http://10.0.10.15/LIF12/`. Sans cela, votre travail ne pourra pas être considéré.

I Transmission de message

Le but de votre programme est de lancer plusieurs processus. Chaque processus dialoguera avec le suivant par un pipe et lui transmettra un message.



I.1 Création des processus

Q.I.1) - Vous devez créer 5 processus père, fils, petit fils, arrière petit fils, arrière^2 petit fils. Attention, il faut faire en sorte que le programme principal ne se termine que lorsque tous les processus se sont terminés.

Q.I.2) - Chaque processus doit partager un pipe avec son fils direct (et seulement avec lui). Le père doit pouvoir écrire dans le pipe et le fils doit pouvoir lire dedans. Les extrémités du pipe non utilisées doivent être correctement fermées.

I.2 Transmission du message

Le premier processus doit générer le premier message et l'envoyer au suivant. Les autres processus doivent lire le message du processus précédant, le traiter l'envoyer au suivant, jusqu'au dernier. Pour la gestion du message, nous avons préparé une fonction :

```
const char* traite_message(int num, const char* message)
```

Elle génère le premier message à envoyer ou traite le message reçu. Attention, ce message est une chaîne de caractères constante qu'il ne faut pas modifier.

Pour générer le premier message, vous pouvez utiliser la fonction traite_message comme cela :

```
const char *mess = traite_message(0, NULL);
```

Pour traiter le message recu (mess_recu) et obtenir le nouveau à transmettre vous pouvez utiliser la fonction comme cela :

```
const char *nouveau_mess = traite_message(num_processus, mess_recu);
```

La fonction traite_message vérifie que le message reçu n'est pas modifié par rapport au message envoyé et affiche un *warning* s'il y a un problème.

Q.I.3) - Programmez la transmission du message en vous assurant que ce dernier sera correctement transmis et lu.

I.3 Redémarrage et fin

Pour le moment, l'ensemble des échanges ne se déroule qu'une seule fois. Vous devez changer cela en faisant en sorte que chaque processus fasse une boucle. Á la fin de chaque boucle, les processus doivent attendre le signal SIGINT (Ctrl-c) ou le signal SIGQUIT (Ctrl-\) :

- lorsqu'il reçoit SIGINT le processus recommence la boucle ;
- lorsqu'il reçoit SIGQUIT le processus termine le programe.

Q.I.4) - Implémentez la gestion des 2 signaux.

Attention, lorsqu'on tape Ctrl-c ou Ctrl-\ dans un terminal, le processus et tous ses descendants reçoivent le signal.

I.4 Résultat d'un processus

Q.I.5) - Faites en sorte que chaque processus récupère le nombre de messages reçus par son fils et l'affiche avant de se terminer.