TP n° 5: πed Piper

L'objectif de ce TP est de vous familiariser avec les *pipe*s en programmation système.

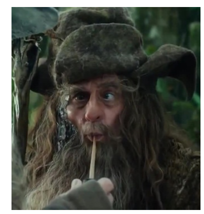
Exercice 1. *Pipe vs. String.*

Nous allons commencer par un exercice simple pour nous familiariser avec les *pipe*s (ou tubes). On veut pouvoir échanger une chaîne de caractères entre deux processus, le père et son fils[1].

1. Commencez par coder un programme simple qui crée un processus à l'aide de `fork` puis s'arrête.
2. On veut échanger de l'information entre ces deux processus. Créez un `pipe` entre le père et le fils puis envoyez une chaîne de caractères du fils vers le père à l'aide de `read` et `write`.

Exercice 2. *Tolkien Ring.*

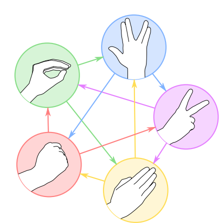
On veut coder une élection dans un anneau, c'est-à-dire décider d'un chef commun. Le père va créer n fils puis un `token` (qui va être le PID du processus) va circuler à travers l'anneau en partant du père. On choisit d'élire le processus au PID le plus élevé. Si un processus reçoit le token mais a un PID plus élevé, il change la valeur du token par son propre PID, puis le passe au suivant.



1. Comment faire pour élire le chef étant donné ce début d'algorithme ?
2. Implémentez une version avec un tableau qui contient tout les `pipes`.
3. Implémentez une version avec seulement 2 `pipes`.

Exercice 3. *Jeux de mains, jeux de vilains*

On veut implémenter un pierre-feuille-ciseaux. Deux processus seront les joueurs, le dernier jouera le rôle d'arbitre. Un fois que chaque joueur envoie son choix à l'arbitre, l'arbitre décide qui a gagné et envoie aux deux joueurs le *PID* du gagnant (pensez à mettre un `sleep` pour que l'on puisse voir quelque chose en console). Le paramètre en entrée n sera le nombre de manche. Le choix entre pierre, feuille et ciseaux sera décidé aléatoirement et indépendamment des manches précédentes.



source : wikimedia

1. Implémentez ce pierre-feuille-ciseaux.
2. Ajoutez un *handler* de signal pour gérer le `Ctrl-C` afin d'arrêter proprement la partie avant la fin des n manches.

1. et le Saint-Esprit

Exercice 4. *Kolakoski*[2].

Soit w un mot, c'est-à-dire une séquence de lettres dans un alphabet. On découpe w en blocs (maximaux) de lettres consécutives identiques. La dérivée w' de w est un mot sur l'alphabet $\mathbb{N}$ tel que $w'[i]$ est la taille du i-ème bloc dans w. Par exemple, la dérivée du mot *abbacccdaab* est le mot 1213121. Le mot de Kolakoski est le mot (infini) sur l'alphabet $\{1, 2\}$, commençant par 1, égal à sa propre dérivée. Il commence donc par 12211212212211..., et la suite est uniquement définie. Ce mot est un casse-tête pour les chercheurs en combinatoire des mots : il n'y a quasiment que des conjectures, sans réussir à trouver des preuves...

1. Écrivez un programme qui prend un mot en entrée, et écrit en sortie sa dérivée.
2. Puis utilisez le principe de deux processus formant une boucle à l'aide de `pipes` pour afficher la suite des préfixes du mot de Kolakoski.
3. *(optionnelle)* Cela bloque rapidement... pourquoi ?

Exercice 5. *L'odysée de Pi.*

Dans cette exercice, le but est de calculer π avec une précision donnée en entrée du programme.
Par exemple :

```
$ ./pi 60
3.141592653589793238462643383279502884197169399375105820974944
```

Pour cela on va utiliser la bibliothèque *mpfr*[3]. On remarque alors que cette bibliothèque possède déjà une fonction `mpfr_const_pi(mpfr_t, mpfr_rnd_t)` qui nous suffit pour faire l'exercice, mais alors il n'aurait pas trop d'intérêt n'est-ce-pas ?
On va donc utiliser la formule de… de qui déjà ? Ha oui, de Machin[4], pour calculer π :

$$\frac{\pi}{4} = 4 \arctan\left(\frac{1}{5}\right) - \arctan\left(\frac{1}{239}\right),$$

en utilisant de développement en série entière de l'arc tangente :

$$\arctan(x) = \sum_{n=0}^{+\infty} \frac{(-1)^n x^{2n+1}}{2n+1}.$$

Quel est le rapport avec le cours demanderiez-vous à votre TD-man favori ? C'est qu'on va faire ça à l'aide de communications interprocessus !

Cet exercice se fera dans un programme `pi.c`.

$$\begin{tikzcd}
 & \text{père} : 4 * (4 * \Sigma_1 + \Sigma_2) \arrow[dl, leftrightarrow] \arrow[dr, leftrightarrow] & \\
\Sigma_1 & & \Sigma_2 \\
\arctan_1 \arrow[u] & & \arctan_2 \arrow[u]
\end{tikzcd}$$

2. /snow/raquettes
3. `http://www.mpfr.org/`
4. Machin… John Machin[5]
5. Si si, c'est bien son nom !

1. Construisez un squelette qui construit le graphe de processus ci-dessus. Autrement dit, le père engendre quatre fils : $\Sigma_1, \Sigma_2, \text{arctan}_1, \text{arctan}_2$, qui vont ouvrir des tubes entre eux comme ceux représentés par les arcs du graphe ci-dessus.
2. Écrivez le contenu des processus $(\text{arctan}_{\mathtt{i}})_{\mathtt{i}\in\{1,2\}}$ qui à la réception du signal `SIGUSR1` vont envoyer à $\Sigma_{\mathtt{i}}$ le terme suivant de la série de l'arc tangente de $\frac{1}{5}$ ou $\frac{1}{239}$ respectivement ; sans tout recalculer !
3. Écrivez le contenu des processus $(\Sigma_{\mathtt{i}})_{i\in\{1,2\}}$, qui vont lire le contenu p de leur tube avec le père et vont calculer la série de l'arc tangente à la précision désirée p en demandant termes à termes les éléments à leur $\text{arctan}_{\mathtt{i}}$ respectifs en utilisant le signal `SIGUSR1`.
4. Écrivez le contenu du père qui va demander à ses fils Σ_i de calculer les arc tangentes à la bonne précision, et qui lira le résultat dans les tubes respectifs, et qui finira par calculer $\pi = 4 \cdot \left(4 \cdot \text{arctan}\left(\frac{1}{5}\right) + \text{arctan}\left(\frac{1}{239}\right)\right)$ et l'afficher à la précision souhaitée en argument.