Exercice 9

Le problème avec les tubes, c'est que des lectures concurrentes du même tube par plusieurs processus donnent des résultats non déterministes : on ne peut pas déterminer quel processus lit quelle donnée. L'exercice de cette semaine a pour but de réaliser un programme pour diriger les lectures par des processus avec des signaux.

Ce programme a la syntaxe suivante :

```
diriger n
```

L'argument n ($n > 0$) correspond au nombre de processus fils à générer (numérotés entre 0 et $n-1$). Le processus père crée un tube unique, puis lit des octets (par bloc) sur l'entrée standard : chaque octet est envoyé individuellement dans le tube, puis le père envoie un signal (par exemple `SIGUSR1`) à un processus fils à tour de rôle pour lui indiquer qu'il peut lire l'octet. L'octet numéro i doit être lu par le fils numéro i mod n et seulement par lui.

Le processus fils numéro k, lorsqu'il ~~reçoit~~ un octet, doit afficher son identité ($0 \leq k < n$), puis l'octet reçu, comme dans l'exemple ci-après :

```
turing> echo abcdefgh | ./diriger 3
0: a
1: b
0: d
2: c
1: e
2: f
0: g
1: h
2:

turing>
```

On respectera le format ci-dessus pour que les tests puissent être réalisés. On notera que le programme doit se terminer lorsqu'il n'y a plus de données sur l'entrée standard, sans intervention manuelle de l'utilisateur. Il est possible que l'ordre ne soit pas strictement respecté (cf octets `d` et `c` dans l'exemple).

Vous utiliserez l'API POSIX des signaux et vous devrez sans doute envoyer des signaux en plus de ceux décrits dans cet énoncé pour que votre programme se comporte conformément aux spécifications. En outre, vous respecterez la règle de bon usage des signaux, c'est-à-dire que vous ferez le minimum d'actions dans la fonction associée à un signal, et vous limiterez l'utilisation des variables globales au strict nécessaire. Bien évidemment, vous n'utiliserez aucune fonction de bibliothèque, excepté les fonctions associées aux signaux, `atoi` et les fonctions d'affichage. Étant donné que plusieurs processus peuvent écrire sur la même sortie standard, vous avez intérêt à utiliser la fonction `fflush` pour ne pas avoir des écritures partiellement effectuées.

Vous trouverez un script de test `test9.sh` sur Moodle.

On rappelle qu'il s'agit d'un travail **individuel**.