TP à rendre 2

Vous devez écrire le programme `ordonnanceur` pour simuler l'ordonnancement des processus sur un monoprocesseur. La syntaxe est la suivante :

```
./ordonnanceur duree_qtum nb_qtm0 [nb_qtm1] [nb_qtm2] ...
```

Ce programme crée plusieurs processus fils puis indique à l'un d'entre eux qu'il peut occuper le processeur. Après expiration de la durée du quantum, l'ordonnanceur (le processus père) évince le processus actuellement sur le processeur et en place un autre. L'ordonnanceur répète les opérations d'élection et d'éviction des processus jusqu'à la terminaison de tous les processus fils. Le programme doit terminer avec un code de retour nul si tous les processus fils ont terminé via un appel à `exit(0)`, ou égal à 1 dans le cas contraire.

La durée d'un quantum est fixée par `duree_qtum` (en secondes). Chaque processus fils a une durée définie en nombre de quanta : `nb_qtm0` définit la durée du processus 0, `nb_qtm1` celle du processus 1, etc. Les processus fils sont sélectionnés par l'ordonnanceur selon un algorithme de type tourniquet.

L'ordonnanceur doit réaliser les affichages suivants en fonction de l'état du système :

```
EVIP - process 0          // le processus 0 a été évincé du processeur
TERM - process 2          // terminaison du processus 2
```

En cas d'élection, un processus doit indiquer sa prise effective du processeur avec le message :

```
SURP - process 3          // le processus 3 est sur le processeur
```

Les processus fils doivent attendre de manière **passive** l'accès au processeur. Cet accès est matérialisé par la réception du signal `SIGUSR1` émis par l'ordonnanceur. Une fois mis sur le processeur, le processus fils doit simuler la tâche à effectuer par une série d'appels à `sleep(1)`. Lorsqu'un quantum expire, l'éviction du processus courant se matérialise par la réception du signal `SIGUSR2` émis par l'ordonnanceur. Vous devez garantir qu'un unique processus se trouve sur le processeur à un instant t, ce qui signifie qu'un processus évincé doit notifier l'ordonnanceur qu'il s'est retiré du processeur via l'émission du signal `SIGUSR1`. L'ordonnancement est gouverné par le signal SIGALRM dont la réception par l'ordonnanceur matérialise la fin d'un quantum. Enfin, un processus qui termine envoie le signal `SIGCHLD` à l'ordonnanceur. Un exemple d'exécution du programme est donné en annexe.

Vous écrirez le programme `ordonnanceur` en langage C en utilisant uniquement les primitives système. La gestion des signaux et la temporisation doivent être réalisées exclusivement avec l'API POSIX des signaux. Les fonctions de bibliothèque sont autorisées uniquement pour l'affichage ou l'allocation mémoire. Pour des raisons d'efficacité, vous ne ferez pas d'appels redondants aux primitives systèmes. Votre programme doit vérifier les valeurs de retour de toutes les primitives système utilisées.

Vous apporterez un soin particulier à la mise en forme de façon à rendre un code lisible et commenté à bon escient. Référez-vous au document « Conseils de programmation pour réussir vos projets et TP » (mis à votre disposition sur Moodle) et, si besoin, utilisez l'utilitaire `clang-format` avec les options données dans ce document.

Votre programme doit compiler avec la commande suivante (disponible dans le `Makefile` mis à disposition sur Moodle) : `cc -Wall -Wextra -Werror`
Les programmes qui ne compilent pas avec cette commande **ne seront pas examinés**. Un script de test est mis à votre disposition sur Moodle. Celui-ci exécute votre programme sur des jeux de tests qui serviront de base à l'évaluation de votre rendu. La commande suivante permet de lancer les tests :
`./test`

N'hésitez pas à contacter votre enseignant si vous constatez un comportement anormal ou si vous souhaitez ajouter un test.

Vous devrez rendre sur Moodle un *unique* fichier nommé `ordonnanceur.c` (Moodle sait qui vous êtes, il est inutile d'appeler votre programme `Jean-Claude_Dusse_ordonnanceur.c`, et il est interdit de rendre un fichier d'un autre nom ou une archive au format du jour). De plus, assurez-vous de rendre des fichiers utilisant l'encodage UTF-8 — il y aura de sévères pénalités sinon.

Ce TP à rendre est **individuel**. On rappelle que la copie ou le plagiat sont sévèrement sanctionnés.

Annexe

Exemple de fonctionnement du programme :

```
./ordonnanceur 1 2 2
SURP - process 0
EVIP - process 0
SURP - process 1
EVIP - process 1
SURP - process 0
EVIP - process 0
SURP - process 1
TERM - process 0
EVIP - process 1
TERM - process 1
```

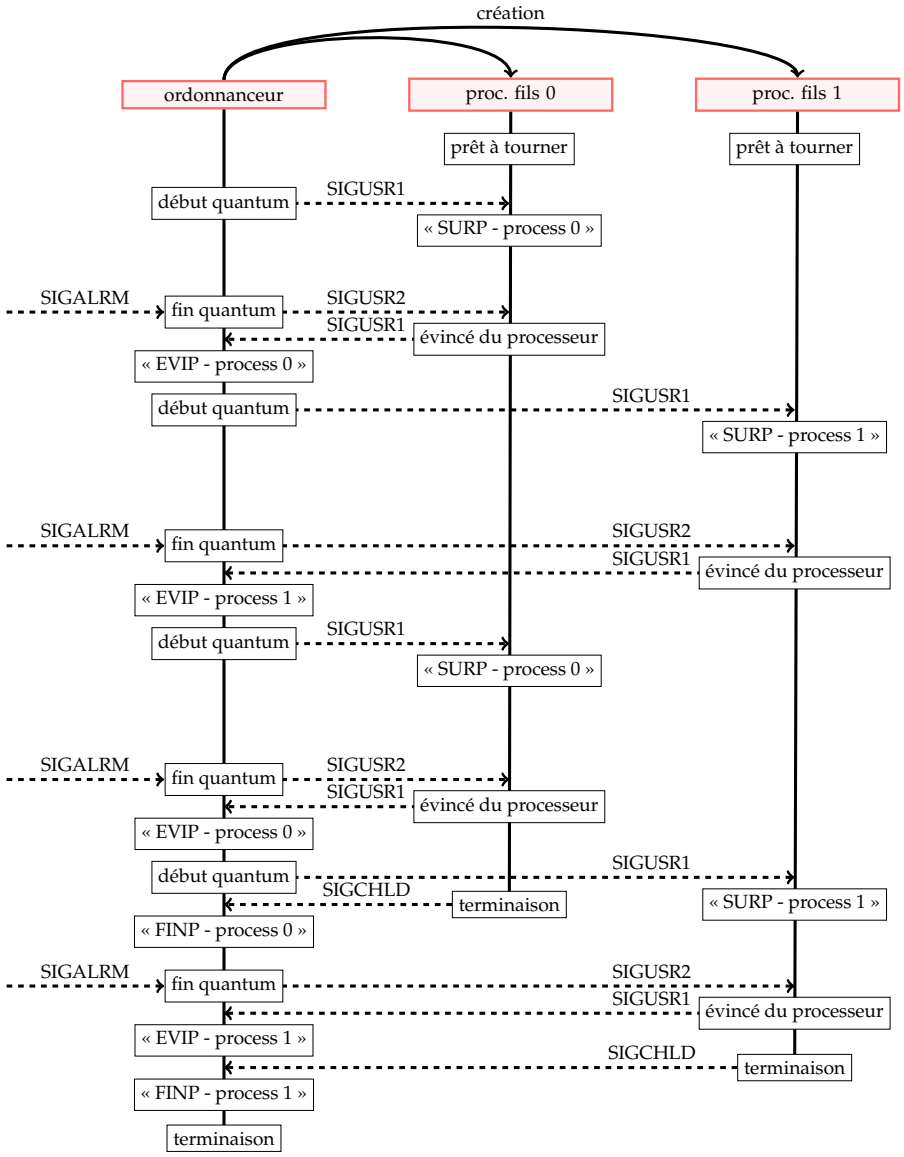

FIGURE 1 – Fonctionnement du programme avec 2 processus fils, une durée de quantum de 1 sec. et la durée de chaque processus fils fixée à 2 quanta