**Explication du code**

Réutilisation du code du dossier PETRA multipièce avec ajout d’une section pour gérer les cas de time-out.

**Critères de l’application** : Gestion des time-outs avec choix de reprendre l’exécution ou de quitter proprement l’application.

**Nouvelles variables** :

Int timeoutOwner : Variable dans laquelle est stocké le tid d’un thread pièce s’il est en train d’exécuter le code du handler HandlerSIGCHLD. Cette variable servira à déterminer quels threads pièce devront recevoir un signal ou non si la section critique d’un time-out est en cours.

Timer\_t timerId : Variable locale à chaque thread pièce qui va contenir l’id d’un timer.

Struct itimerspec timerOut : Instance de structure itimerspec utilisée pour initialiser un timer.

**Nouvelles sections du code** :

La mise en marche des timers :

Un timer est lancé dans chaque thread pièce avant d’entrer dans la boucle d’attente d’une pièce au niveau du slot ou du corner. Si une pièce arrive avant la fin du timer, celui-ci est réinitialisé à 0. Si aucune pièce n’arrive avant la fin du timer, un SIGCHLD est envoyé et le code du HandlerSIGCHLD est exécuté par le thread interrompu.

Le handler SIGCHLD :

Le mutex mutexTimeout est verrouillé, les convoyeurs sont arrêtés et un signal SIGUSR2 est envoyé à tous les threads sauf celui-ci dont le tid est égal au timeoutOwner. Les autres threads restent ainsi bloqués dans le HandlerSIGUSR2 tant que la section critique timeout est en cours.

L’utilisateur a ensuite le choix entre reprendre l’exécution ou quitter l’application. Si l’exécution est reprise, les convoyeurs sont redémarrés et mutexTimeout est déverrouillé. Si l’application est quittée, on supprime les ressources avant de quitter le programme via un exit.