PC3R - TME2: Producteurs / Consommateurs Coopératifs

Equipe Enseignante PC(2/3)R

10/02/2022

Lien vers l'API Fair Threads en C: `https://www-sop.inria.fr/mimosa/rp/FairThreads/FTC/`

Objectif: Manipuler des processus évoluant avec des ordonnanceurs prévisibles.

Rendu: Le rendu final pour ce TME doit être une unique archive contenant les sources qui peuvent être, au choix, un unique fichier `.c` ou un répertoire avec un `Makefile`.

Evaluation: Le rendu est évalué sur:
- la pertinence de l'utilisation des mécanismes proposés par l'API des *fair threads*.
- la lisibilité du code.
- la justesse du choix des paramètres (endormissement temporaire des threads, taille des tapis, nombre de producteurs et de consommateurs), permettant de démontrer le fonctionnement concurrent du système.

Description du Système

Le threads du système sont divisés ainsi:
- un thread principal, qui crée les tapis et les autres threads et attend leur terminaison,
- n threads *producteurs* qui fabriquent des paquets et les enfilents sur le tapis de production,
- m threads *consommateurs* qui consomment les paquets depuis le tapis de consommation.
- p threads *messagers* qui transportent des paquets du tapis de production au tapis de consommation

Les données du systèmes sont organisées ainsi:
- les *paquets* sont des structures C (`struct`) qui encapsulent une unique chaîne de caractères.
- les *tapis* sont des structures C qui contiennent une file de paquet, et une capacité. Il ne peut y avoir un nombre de paquet dans la file du tapis strictement plus grand que sa capacité.
- les *tapis* disposent de procédures / méthodes permettant à un thread d'*enfiler* un paquet. Si le tapis n'est pas plein, le paquet est mis en bout de file, sinon, le thread attend que le tapis soit non-plein avant de réessayer.
- les *tapis* disposent d'une procédure / méthode permettant à un thread de *défiler* un paquet. Si le tapis n'est pas vide, le paquet en tête de file est retiré et renvoyé au thread comme valeur de retour de la procédure / méthode, sinon, le thread attend que le tapis soit non-vide avant de réessayer.
- un *journal* de trajet est écrit par les messagers.
- deux *journaux* de production et de consommation sont écrits par (respectivement) les producteurs et les consommateurs.

Producteurs Les threads producteurs sont initialisés avec un nom de produit (différents pour chaque producteur). Ils font référence à un journal de production partagé, et utilisent une cible de production entière (une constante). Leur comportement est donné par:
- chaque producteur est attaché à l'**ordonnanceur de production**,
- chaque producteur tourne en boucle tant qu'il a produit moins de paquet que sa cible de production. A chaque tour de boucle:
  - il crée un nouveau paquet avec comme contenu le nom du produit associé concaténé à un entier comptant le nombre de produits déjà créés (par exemple `"Pomme 3"`)

  - il enfile le paquet dans le tapis en utilisant la procédure décrite plus haut. Si le tapis de production est plein, il s'endort jusqu'à être réveillé, il réessaye ensuite d'enfiler.
  - il enregistre le succès de son enfilage dans le journal de production.
  - il coopère.

Consommateurs Les threads consommateurs sont initialisés avec un identifiant entier. Ils font référence à un compteur de consommation partagé et au journal de consommation partagé. Leur comportement est donné par:

- chaque consommateur est attaché à l'**ordonnanceur de consommation**,
- chaque consommateur tourne en boucle tant que le compteur est supérieur à 0. A chaque tour de boucle:
  - il défile un paquet du tapis. Si le tapis est vide, il s'endort jusqu'à être réveillé, il essaye ensuite de défiler.
  - il enregistre le succès de son défilage dans le journal de consommation.
  - il décrémente le compteur.
  - il coopère.

Messagers Les threads messagers sont initialisés avec un identifiant entier. Ils font référence à un journal de voyage partagé et connaissent les deux ordonnanceurs (production / consommation).

- initialement un thread messager est *vide* et n'est attaché à *aucun* ordonnanceur.
- chaque messager tourne en boucle tant que le compteur de consommation partagé n'est pas 0. A chaque tour de boucle:
  - il se lie à l'ordonnanceur de production.
  - il essaye de défiler le tapis de production, il s'endort si ce dernier est vide, et essaye à nouveau quand il est réveillé.
  - une fois qu'il a réussi à récupérer un paquet, il se détache de l'ordonnanceur de production.
  - il écrit son trajet dans le journal de voyage.
  - il se lie à l'ordonnanceur de consommation.
  - il essaye d'enfiler son paquet dans l'ordonnanceur de consommation, il s'endort si ce dernier est plein, et essaye à nouveau quand il est réveillé.
  - une fois qu'il a réussi à enfiler son paquet, il se détache de l'ordonnanceur de consommation

Thread principal Le thread principal initialise le tapis, le compteur à une valeur égale à la cible de production des producteurs multipliée par le nombre de producteurs, et lance les producteurs et les consommateurs. Il attend ensuite que le compteur arrive à zéro, puis termine.