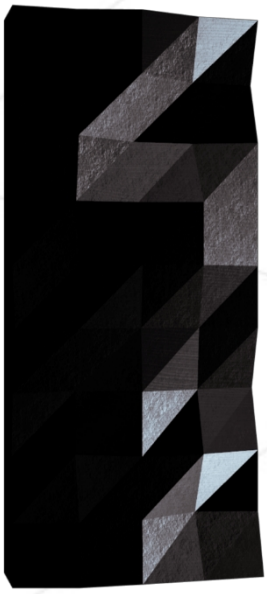

Philosophes

Je n'aurais jamais pensé que la philosophie puisse être aussi mortelle

Résumé :
Dans ce projet, vous apprendrez les bases du threading d'un processus. Vous verrez comment créer des threads et vous découvrirez les mutex.

Version : 11

Contenu

Chapitre I

Introduction

La philosophie (du grec *philosophia*, littéralement "amour de la sagesse") est l'étude des questions générales et fondamentales concernant l'existence, la connaissance, les valeurs, la raison, l'esprit et la langue. Ces questions sont souvent posées comme des problèmes à étudier ou à résoudre. Le terme a probablement été inventé par Pythagore (vers 570 - 495 avant J.-C.). Les méthodes philosophiques comprennent le questionnement, la discussion critique, l'argumentation rationnelle et la présentation systématique.

Les questions philosophiques classiques sont les suivantes : Est-il possible de savoir quelque chose et de le prouver ? Qu'est-ce qui est le plus réel ? Les philosophes posent également des questions plus pratiques et concrètes telles que : Existe-t-il une meilleure façon de vivre ? Est-il préférable d'être juste ou injuste (si l'on peut s'en sortir) ? Les êtres humains ont-ils un libre arbitre ?

Historiquement, la "philosophie" englobe tout ensemble de connaissances. De l'époque du philosophe grec antique Aristote jusqu'au XIXe siècle, la "philosophie naturelle" englobait l'astronomie, la médecine et la physique. Par exemple, les Principes mathématiques de la philosophie naturelle de Newton, publiés en 1687, ont été classés plus tard comme un livre de physique.

Au 19e siècle, la croissance des universités de recherche modernes a conduit la philosophie et d'autres disciplines à se professionnaliser et à se spécialiser. À l'époque moderne, certaines recherches qui faisaient traditionnellement partie de la philosophie sont devenues des disciplines universitaires distinctes, notamment la psychologie, la sociologie, la linguistique et l'économie.

D'autres recherches étroitement liées à l'art, à la science, à la politique ou à d'autres activités sont restées dans le domaine de la philosophie. Par exemple, la beauté est-elle objective ou subjective ? Existe-t-il plusieurs méthodes scientifiques ou une seule ? L'utopie politique est-elle un rêve plein d'espoir ou un fantasme sans espoir ? Les principaux sous-domaines de la philosophie universitaire comprennent la métaphysique ("qui s'intéresse à la nature fondamentale de la réalité et de l'être"), l'épistémologie (qui porte sur "la nature et les fondements de la connaissance [et]... ses limites et sa validité"), l'éthique, l'esthétique, la philosophie politique, la logique et la philosophie des sciences.

Chapitre II

Instructions communes

- Votre projet doit être écrit en C.
- Votre projet doit être rédigé conformément à la norme. Si vous avez des fichiers/fonctions bonus, ils sont inclus dans le contrôle de la norme et vous recevrez un 0 s'il y a une erreur de norme à l'intérieur.
- Vos fonctions ne doivent pas se terminer de manière inattendue (erreur de segmentation, erreur de bus, double free, etc) en dehors des comportements non définis. Si cela se produit, votre projet sera considéré comme non fonctionnel et recevra un 0 lors de l'évaluation.
- Tout l'espace mémoire alloué au tas doit être libéré correctement lorsque cela est nécessaire. Aucune fuite ne sera tolérée.
- Si le sujet l'exige, vous devez soumettre un Makefile qui compilera vos fichiers sources vers la sortie requise avec les drapeaux -Wall, -Wextra et -Werror, utiliser cc, et votre Makefile ne doit pas être relié.
- Votre Makefile doit au moins contenir les règles $(NAME), all, clean, fclean et re.
- Pour ajouter des bonus à votre projet, vous devez inclure une règle bonus à votre Makefile, qui ajoutera tous les différents headers, librairies ou fonctions qui sont interdits sur la partie principale du projet. Les bonus doivent être dans un fichier différent _bonus.{c/h} si le sujet ne spécifie rien d'autre. L'évaluation des parties obligatoires et bonus se fait séparément.
- Si votre projet vous permet d'utiliser votre libft, vous devez copier ses sources et son Makefile associé dans un dossier libft avec son Makefile associé. Le Makefile de votre projet doit compiler la bibliothèque en utilisant son Makefile, puis compiler le projet.
- Nous vous encourageons à créer des programmes de test pour votre projet, même si ce travail **ne devra pas être présenté et ne sera pas noté**. Cela vous permettra de tester facilement votre travail et celui de vos camarades. Ces tests vous seront particulièrement utiles lors de votre soutenance. En effet, lors de la soutenance, vous êtes libre d'utiliser vos tests et/ou les tests du pair que vous évaluez.
- Soumettez votre travail dans le dépôt git qui vous a été attribué. Seul le travail dans le dépôt git sera noté. Si Deepthought est chargé de noter votre travail, il le fera.

après l'évaluation de vos pairs. Si une erreur se produit dans une section de votre travail pendant l'évaluation de Deepthought, l'évaluation s'arrêtera.

Chapitre III Vue d'ensemble

Voici ce qu'il faut savoir pour réussir cette mission :

- Un ou plusieurs philosophes sont assis à une table ronde.
  Un grand bol de spaghettis se trouve au milieu de la table.
- Les philosophes **mangent**, **pensent** ou **dorment** alternativement.
  Pendant qu'ils mangent, ils ne pensent ni ne dorment ; pendant
  qu'ils pensent, ils ne mangent ni ne dorment ;
  et, bien sûr, pendant leur sommeil, ils ne mangent pas et ne pensent pas.
- Il y a aussi des fourchettes sur la table. Il y a **autant de fourchettes que de philosophes**.
- Parce qu'il n'est pas très pratique de servir et de manger des spaghettis avec une seule fourchette, un philosophe prend sa fourchette droite et sa fourchette gauche pour manger, une dans chaque main.
- Lorsqu'un philosophe a fini de manger, il repose sa fourchette sur la table et commence à dormir. Une fois réveillé, il recommence à réfléchir. La simulation s'arrête lorsqu'un philosophe meurt de faim.
- Tout philosophe a besoin de manger et ne devrait jamais mourir de faim.
- philosophes ne se parlent pas entre .
- Les philosophes ne savent pas si un autre philosophe est sur le point de mourir.
- Inutile de dire que les philosophes doivent éviter de mourir !

Chapitre IV Règles mondiales

Vous devez écrire un programme pour la partie obligatoire et un autre pour la partie bonus (si vous décidez de faire la partie bonus). Tous deux doivent respecter les règles suivantes :

- Les variables globales sont interdites !
- Votre (vos) programme(s) doit (doivent) prendre les arguments suivants : nombre_de_philosophes temps_de_mourir temps_de_manger temps_de_dormir [nombre_de_fois_que_chaque_philosophe_doit_manger].
  - nombre_de_philosophes : Le nombre de philosophes et aussi le nombre de fourches.
  - time_to_die (en millisecondes) : Si un philosophe n'a pas commencé à manger time_to_die millisecondes depuis le début de son dernier repas ou le début de la simu- lation, il meurt.
  - time_to_eat (en millisecondes) : Le temps qu'il faut à un philosophe pour manger. Pendant ce temps, il devra tenir deux fourchettes.
  - time_to_sleep (en millisecondes) : Le temps qu'un philosophe passera à dormir.
  - nombre_de_fois_que_chaque_philosophe_doit_manger (argument facultatif) : Si tous les philosophes ont mangé au moins le nombre de fois où chaque philosophe doit manger, la simulation s'arrête. Si elle n'est pas spécifiée, la simulation s'arrête lorsqu'un philosophe meurt.
- Chaque philosophe a un numéro allant de 1 à nombre_de_philosophes.
- Le philosophe numéro 1 est assis à côté du philosophe numéro_de_philosophes. Tout autre philosophe numéro N est assis entre le philosophe numéro N - 1 et le philosophe numéro N + 1.

A propos des logs de votre programme :

- Tout changement d'état d'un philosophe doit être formaté comme suit :
  - timestamp_in_ms X a pris une fourche
  - timestamp_in_ms X mange
  - timestamp_in_ms X est en train de dormir
  - timestamp_in_ms X pense
  - horodatage_in_ms X mort

  Remplacez timestamp_in_ms *par le timestamp actuel en millisecondes et* X *par le nombre de philosophes.*
- Un message d'état affiché ne doit pas être confondu avec un autre message.
- Un message annonçant le décès d'un philosophe ne doit pas être affiché plus de 10 ms après la mort effective du philosophe.
- Encore une fois, les philosophes devraient éviter de mourir !

Votre programme ne doit pas comporter de **course aux données.**

Chapitre V Partie obligatoire

Nom du programme	philo
Remettre les dossiers	Makefile, *.h, *.c, dans le répertoire philo/
Makefile	NOM, tous, nettoyer, fclean, re
Arguments	nombre_de_philosophes temps_de_mourir temps_de_manger temps_de_dormir [nombre_de_fois_que_chaque_philosophe_doit_manger]
Fonctions externes.	memset, printf, malloc, free, write, usleep, gettimeofday, pthread_create, pthread_detach, pthread_join, pthread_mutex_init, pthread_mutex_destroy, pthread_mutex_lock, pthread_mutex_unlock
Libft autorisé	Non
Description	Des philosophes avec des threads et des mutex

Les règles spécifiques pour la partie obligatoire sont les suivantes :

- Chaque philosophe devrait être un fil conducteur.
- Il y a une fourchette entre chaque paire de philosophes. Par conséquent, s'il y a plusieurs philosophes, chaque philosophe a une fourchette à sa gauche et une fourchette à sa droite. S'il n'y a qu'un seul philosophe, il ne doit y avoir qu'une seule fourchette sur la table.
- Pour empêcher les philosophes de dupliquer les forks, vous devez protéger l'état des forks avec un mutex pour chacun d'entre eux.

Chapitre VI Partie bonus

Nom du programme	philo_bonus
Remettre les dossiers	Makefile, *.h, *.c, dans le répertoire philo_bonus/
Makefile	NOM, tous, nettoyer, fclean, re
Arguments	nombre_de_philosophes temps_de_mourir temps_de_manger temps_de_dormir [nombre_de_fois_que_chaque_philosophe_doit_manger]
Fonctions externes.	memset, printf, malloc, free, write, fork, kill, exit, pthread_create, pthread_detach, pthread_join, usleep, gettimeofday, waitpid, sem_open, sem_close, sem_post, sem_wait, sem_unlink
Libft autorisé	Non
Description	Des philosophes avec des processus et des sémaphores

Le programme de la partie bonus prend les mêmes arguments que le programme obligatoire. Il doit être conforme aux exigences du chapitre "*Règles générales*".

Les règles spécifiques pour la partie bonus sont les suivantes :

- Toutes les fourchettes sont placées au milieu de la table.
- Ils n'ont pas d'état en mémoire mais le nombre de fourches disponibles est représenté par un sémaphore.
- Chaque philosophe devrait être un processus. Mais le processus principal ne doit pas être un philosophe.

La partie bonus ne sera évaluée que si la partie obligatoire est PARFAITE. Parfait signifie que la partie obligatoire a été intégralement réalisée et qu'elle fonctionne sans dysfonctionnement. Si vous n'avez pas satisfait à TOUTES les exigences obligatoires, votre partie bonus ne sera pas évaluée du .

Chapitre VII

Soumission et évaluation par les pairs

Rendez votre travail dans votre dépôt Git comme d'habitude. Seul le travail contenu dans votre dépôt sera évalué lors de la soutenance. N'hésitez pas à vérifier les noms de vos fichiers pour vous assurer qu'ils sont corrects.

Répertoire des parties obligatoires : philo/

Répertoire de la partie bonus : philo_bonus/