哲学者

哲学がこれほどまでに命取りになるとは

概要
このプロジェクトでは、プロセスのスレッド化の基本を学びます。スレッドを作成する方法と、ミューテックスを発見することができます。

バージョン：10

内容

第一章 はじめに

哲学は、存在、知識、価値、理性、心、言語などに関する一般的かつ根本的な問題を研究する学問である。このような疑問は、しばしば研究または解決されるべき問題として提起される。この言葉はおそらくピタゴラス（紀元前570年頃-紀元前495年頃）によって作られた。哲学の方法には、問いかけ、批判的議論、合理的な議論、体系的な提示などがある。

古典的な哲学の問いには、次のようなものがあります。何かを知り、それを証明することは可能か？最も現実的なものは何か？また、哲学者は、より実践的で具体的な次のような問いを投げかけます。生きるための最良の方法はあるか？生きるための最良の方法はあるのか、正義と悪のどちらがよいのか（もしそれが可能なら）。人間には自由意志があるのか？

歴史的に見れば、「哲学」はあらゆる知識体系を包含している。古代ギリシャの哲学者アリストテレスの時代から19世紀まで、「自然哲学」は天文学、医学、物理学を包含していた。例えば、ニュートンが1687年に発表した『自然哲学の数学的原理』は、後に物理学の書物として分類されるようになった。

19世紀には、近代的な研究型大学の発展により、哲学をはじめとする学問は専門化し、特化するようになりました。近代になると、心理学、社会学、言語学、経済学など、伝統的に哲学の一部であった学問が独立した学問分野となったものもある。

また、芸術、科学、政治などに密接に関連する研究は、哲学の一部として残りました。例えば、美は客観的か主観的か？科学的な方法はたくさんあるのか、それともひとつだけなのか。政治的ユートピアは希望に満ちた夢なのか、それとも絶望的な空想なのか？哲学の主な分野としては、形而上学（「現実と存在の楽しいダム的性質に関する」）、認識論（「知識の性質と根拠（および）...その限界と有効性」に関する）、倫理学、美学、政治哲学、論理学、科学哲学などがある。

第二章

共通事項

- プロジェクトはC言語で書かれている必要があります。
- あなたのプロジェクトはNormに則って書かれていなければなりません。ボーナスファイル/関数がある場合、それらはノームチェックに含まれ、内部にノームエラーがある場合は0が返されます。
- 未定義の動作とは別に、関数が予期せず終了する (セグメンテーションフォールト、バスエラー、ダブルフリーなど) ことがないようにしてください。このような場合、プロジェクトは非機能とみなされ、評価時に0点が与えられます。
- ヒープで確保されたメモリ空間は、必要なときに適切に解放されなければなりません。リークは許されません。
- 対象がそれを要求している場合、ソースファイルを要求された出力にコンパイルする Makefile を -Wall、-Wextra、-Werror フラグで提出し、cc を使用し、その Makefile は再リンクしてはいけません。
- Makefileには、少なくとも$(NAME)、all、clean、fclean、および
  再
- プロジェクトにボーナスを回すには、Makefile にルールボーナスを含める必要があり、 それによってプロジェクトのメイン部分で禁止されている様々なヘッダー、librairies または関数がすべて追加されます。ボーナスは、主体が何も指定しない場合は、別のファイル _bonus.{c/h} にする必要があります。必須部分とボーナス部分の評価は別々に行われます。
- プロジェクトで libft を使用できるようにする場合、そのソースと関連する Makefile を libft フォルダにコピーしておく必要があります。あなたのプロジェクトの Makefile は、その Makefile を使ってライブラリをコンパイルし、 その後プロジェクトをコンパイルしなければなりません。
- テストプログラムは提出**する必要がなく、採点もされませんが**、プロジェクトのテストプログラムを作成することをお勧めします。テストプログラムを作成することで、自分の作品や仲間の作品を簡単にテストすることができます。このようなテストは、特にデフェンスの時に役立つと思います。実際、審査では、自分のテストや審査する仲間のテストを自由に使用することができます。
- 指定された git リポジトリに作品を提出する。git リポジトリにある作品だけが採点されます。Deepthought があなたの作品の採点を担当する場合、採点は以下のように行われます。

相互評価後Deepthoughtの採点中に、あなたの作品のいずれかのセクションでエラーが発生した場合、評価は停止されます。

第III章 概要

ここでは、この課題を成功させるために知っておくべきことを紹介します。

- 一人または複数の哲学者が円卓に座っている。
  テーブルの真ん中には、スパゲッティの入った大きなボウルが置いてある。
- 哲学者たちは、食べたり、考えたり、**眠ったりすることを繰り返している**。食べている間は、考えることも眠ることもなく、考えている間は、食べることも眠ることもない。
  もちろん、寝ている間は、食べたり、考えたりしているわけではありません。
- テーブルの上にはフォークもあります。**哲学者の数だけフォークが**あるのです。
- スパゲッティをフォーク1本で食べるのはとても不便なので、哲学者は右手と左手に1本ずつフォークを持って食べます。
- 哲学者は食事を終えると、フォークをテーブルに戻し、眠り始める。目が覚めると、再び思考を開始する。哲学者が餓死すると、シミュレーションは終了する。
- すべての哲学者は食べる必要があり、決して飢えてはならない。
- 哲学者同士は話をしない。
- 哲学者は、他の哲学者が死のうとしているのかどうかを知らない。
- 哲学者は死ぬのを避けるべきと言う必要はない!

第IV章 グローバ

ルルール

あなたは、必須パート用のプログラムと、ボーナスパート用のプログラム（ボーナスパートを行うことにした場合）を作成する必要があります。どちらも以下のルールに従わなければならない。

- グローバル変数の使用は禁止されています。
- あなたの(複数の)プログラムは次の引数を取るべきである:
  number_of_philosophers time_to_die time_to_eat time_to_sleep
  [number_of_times_each_philosopher_must_eat] [引数].
  - number_of_philosophers。哲学者の数、またフォークの数。
  - time_to_die (in milliseconds):もし哲学者が最後の食事を始めてから、あるいはシミュレーションを始めてから time_to_die ミリ秒経過しても食事を始めなかった場合、その哲学者は死にます。
  - time_to_eat (ミリ秒)。哲学者が食事をするのにかかる時間。その間、彼らは2本のフォークを持つ必要がある。
  - time_to_sleep (ミリ秒):哲学者が睡眠に費やす時間。
  - number_of_times_each_philosopher_must_eat (オプションの引数):すべての哲学者が少なくとも number_of_times_each_philosopher_must_eat 回食べれば, シミュレーションは 停止指定しない 場合は哲学者が死亡した時点でシミュレーションを停止します。
- 各哲学者は1からnumber_of_philosophersまでの番号を持っています。
- 哲学者番号1は哲学者番号number_of_philosophersの隣に座る。他の哲学者番号Nは、哲学者番号N - 1と哲学者番号N + 1の間に位置する。

番組のログについて

- フィロソフィーの状態変化は、以下のような書式にする必要があります。
  - timestamp_in_ms X がフォークを取得しました。
  - timestamp_in_ms Xは食事中です。
  - timestamp_in_ms Xがスリープ状態であること
  - timestamp_in_ms Xは考えています。
  - timestamp_in_ms X 死亡

  timestamp_in_ms を*ミリ秒単位の現在のタイムスタンプに*、X *を哲学者番号に置き換えてください。*
- 表示された状態メッセージは、他のメッセージと混在させてはならない。
- 哲学者の死を告げるメッセージは、実際の死から10ミリ秒以内に表示される必要があります。
- 繰り返すが、哲学者は死なない方がいいのだ

プログラムには、**データレースがあってはならない。**

第五章 必須項目

プログラム名	フィロ
ファイルを提出する	Makefile, *.h, *.c, in directory philo/
メイクファイル	NAME、All、 Clean、Fclean、Re
論証	哲学者の数 死ぬまでの時間 食べるまでの時間 time_to_sleep [number_of_times_each_philosopher_must_eat] (各フィロソフィーの必食回数)
外部機能。	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
リブート認可	いいえ
商品説明	スレッドとミューテックスを使った哲学者たち

必須部分の具体的なルールは

- 哲学者一人一人が糸になること。
- 各対の哲学者の間にはフォークが1本ずつある。したがって、哲学者が何人かいる場合、それぞれの哲学者は左側にフォークを持ち、右側にフォークを持つ。哲学者が一人しかいない場合、テーブルの上にはフォークは一本しかないはずです。
- 哲学者のフォークの重複を防ぐため、フォークの状態をそれぞれミューテックスで保護する必要があります。

第VI章 ボーナス

パート

プログラム名	フィロ・ボーナス
ファイルを提出する	ディレクトリ philo_bonus/ の Makefile, *.h, *.c, にあります。
メイクファイル	NAME、All、 Clean、Fclean、Re
論証	哲学者の数 死ぬまでの時間 食べるまでの時間 time_to_sleep [number_of_times_each_philosopher_must_eat] (各フィロソフィーの必食回数)
外部機能。	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
リブート認可	いいえ
商品説明	プロセスやセマフォを持つ哲学者たち

ボーナスパートのプログラムは、必須プログラムと同じ引数を取ります。
*グローバルルール*の章の要件に準拠する必要があります。

ボーナスパートの具体的なルールは

- すべてのフォークはテーブルの真ん中に置かれる。
- これらはメモリ上に状態を持たないが、利用可能なフォークの数はセマフォで表現される。
- 各哲学者は プロセスであるべきだ。 しかし主要なプロセスは哲学者であってはならない。

ボーナスパーツは、必須パーツがPERFECTである場合にのみ査定されます。パーフェクトとは、必須パートが統合的に行われ、誤動作することなく動作することを意味します。 必須条件をすべてクリアしていない場合、ボーナスパーツの評価は一切行われません。

第七章

提出と相互評価

通常通り、Git リポジトリに課題を提出してください。レポ-ト内の作品だけが、ディフェンスで評価されます。ファイル名が正しいかどうか、遠慮なく再確認してください。

必須パートディレクトリ：philo/

ボーナス部ディレクトリ: philo_bonus/