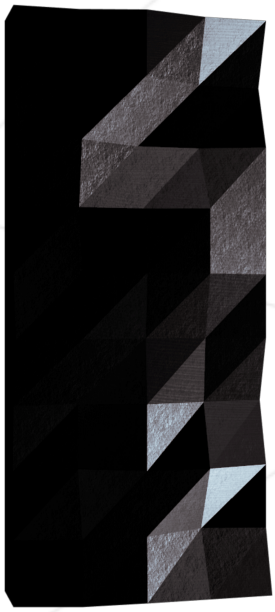

Philosophers

Quién diría que la filosofía es tan poco vívida.

Resumen: En este proyecto, entenderás los principios del threading a un proceso. Aprenderás a hacer threads. Descubrirás los mutex.

Índice general

I. Introducción **2**

II. Parte obligatoria **3**

III. Bonus **6**

Capítulo I

Introducción

La filosofía (del griego antiguo “philosophia” o “amor a la sabiduría”), es una disciplina académica y un conjunto de reflexiones y conocimientos de carácter trascendental que, en un sentido holístico, estudia la esencia, las causas primeras y los fines últimos de las cosas. Trata de responder a una variedad de problemas fundamentales acerca de cuestiones como la existencia y el ser (ontología y metafísica), el conocimiento (epistemología y gnoseología), la verdad (lógica), la moral (ética), la belleza (estética), el valor (axiología), la mente (filosofía de la mente), el lenguaje (filosofía del lenguaje) y la religión (filosofía de la religión). A lo largo de la historia, muchas otras disciplinas han surgido a raíz de la filosofía, por lo que es considerada la base de todas las ciencias modernas por muchos autores. El término probablemente fue acuñado por Pitágoras.

La filosofía occidental ha influido sobre otras ramas del conocimiento humano, por ejemplo, en el ámbito de la ciencia, la religión y la política. Muchos filósofos importantes fueron a la vez grandes científicos, teólogos o políticos y algunas nociones fundamentales de estas disciplinas todavía son objeto de estudio filosófico. Esta superposición entre disciplinas se debe a que la filosofía es una disciplina muy amplia. En el siglo XIX, el crecimiento de las universidades de investigación modernas llevó a la filosofía académica y otras disciplinas a profesionalizarse y especializarse. Desde entonces, varias áreas de investigación que tradicionalmente formaban parte de la filosofía se han convertido en disciplinas académicas separadas, como la psicología, la sociología, la biología, la lingüística y la economía.

Capítulo II

Parte obligatoria

Tendrás que escribir un programa para la parte obligatoria y un programa para la parte bonus. Ambos seguirán los mismos principios:

- Este proyecto se debe programar en C, siguiendo la Norma. Cualquier leak, crash, comportamiento indefinido, o error de Norma significa un 0.
- Varios filósofos se sientan en una mesa redonda haciendo una de estas tres cosas: comer, pensar o dormir.
- Mientras comen, no piensan ni duermen. Mientras duermen, ni comen ni piensan y, por supuesto, mientras piensan no comen o duermen.
- Los filósofos se sientan en una mesa circular con un gran bol de espaguetis en el centro.
- Hay algunos tenedores en la mesa.
- Dado que los espaguetis son difíciles de servir y comer con un solo tenedor, se asume que un filósofo debe comer con dos tenedores, uno por cada mano.
- Los filósofos nunca deben estar hambrientos.
- Cada filósofo debe comer.
- Los filósofos no hablan entre ellos.
- Los filósofos no saben cuando otro filósofo va a morir.
- Cada vez que un filósofo ha terminado de comer, dejará los tenedores y empezará a dormir.
- Cuando un filósofo termina de dormir, empezará a pensar.
- La simulación termina cuando un filósofo muere.
- Cada programa debe aceptar el mismo número de opciones: número_de_filósofos tiempo_para_morir tiempo_para_comer tiempo_para _dormir [número_de_veces_que_cada_filósofo_debe_comer]
    - `número_de_filósofos`: es el número de filósfos y también el número de tenedores.

- `tiempo_para_morir`: en milisegundos, si un filósofo no empieza a comer `tiempo_para_morir` milisegundos después de su última comida o de la simulación, muere.
- `tiempo_para_comer`: en milisegundos, el tiempo que usa cada filósofo para comer. Durante ese tiempo usará dos tenedores.
- `tiempo_para_dormir`: en milisegundos, el tiempo que un filósofo pasa durmiendo.
- `número_de_veces_que_cada_filósofo_debe_comer`: este argumento es opcional, si todos los filósofos comen al menos "`número_de_veces_que_cada_filósofo_debe_comer`" la simulación terminará. Si no se especifica, la simulación terminará solo tras la muerte de un filósofo.

- Cada filósofo debe recibir un número del 1 al "`número_de_filósofos`".
- El filósofo número 1 está junto al filósofo "`número_de_filósofos`". Cualquier otro filósofo con el número N está sentado entre el filósofo N 1 y el N + 1.
- Cualquier cambio de estado de un filósofo debe mostrarse de la siguiente manera (reemplaza X por el número de filósofo y `timestamp_en_ms` por el timestamp actual en milisegundos):
  - `timestamp_en_ms` X has taken a fork
  - `timestamp_en_ms` X is eating
  - `timestamp_en_ms` X is sleeping
  - `timestamp_en_ms` X is thinking
  - `timestamp_en_ms` X died
- El estado impreso no debe estar descoordinado o intercambiado con el estado de otro filósofo.
- No puedes tener más de 10 ms entre la muerte de un filósofo y cuándo aparece su muerte.
- De nuevo, evita que los filósofos mueran.

Nombre de programa	`philo`
Archivos a entregar	`philo/`
Makefile	`Sí`
Argumentos	`número_de_filósofos tiempo_para_morir tiempo_para_comer tiempo _para_dormir [número_de_veces_que_cada_filósofo_debe_comer]`
Funciones autorizadas	`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`
Se permite usar libft	`No`
Descripción	`filósofos con threads y mutex`

En esta versión las reglas específicas son:

- Un tenedor entre cada filósofo, por lo que si hay múltiples filósofos, habrá un tenedor a la derecha y otro a la izquierda de cada uno.
- Para evitar que los filósofos dupliquen tenedores, deberás proteger el estado de los tenedores con un mutex por cada uno de ellos.
- Cada filósofo será un hilo.

Capítulo III

Bonus

Nombre de programa	`philo_bonus`
Archivos a entregar	`philo_bonus/`
Makefile	`Sí`
Argumentos	`número_de_filósofos tiempo_para_morir tiempo_para_comer tiempo _para_dormir [número_de_veces_que_cada_filósofo_debe_comer]`
Funciones autorizadas	`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`
Se permite usar libft	`No`
Descripción	`filósofos con procesos y semáforos`

En esta versión las reglas específicas son:

- Todos los tenedores están en el centro de la mesa.
- No tienen estados en memoria pero el número de tenedores disponibles se representa con un semáforo.
- Cada filósofo debe ser un proceso y el proceso principal no debe ser un filósofo.