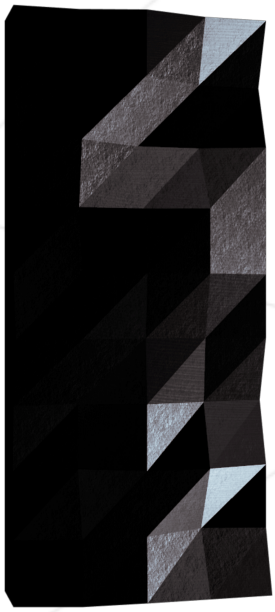

Philosophers

Quién pudiera imaginar que la filosofía fuera tan letal.

Resumen: En este proyecto, aprenderás los principios de hilar un proceso. Aprenderás a hacer hilos. Descubrirás los mutex.

Versión: 9

Índice general

I.	**Introducción**	**2**
II.	**Instrucciones generales**	**3**
III.	**Descripción general**	**4**
IV.	**Instrucciones generales**	**5**
V.	**Parte obligatoria**	**7**
VI.	**Bonus**	**8**

Capítulo I

Introducción

La filosofía (del griego, philosophia, literalmente .ªmor por el saber") es el estudio de preguntas generales y fundamentales sobre la existencia, el saber, los valores, la razón, la mente, y el lenguaje. Algunas preguntas se tratan como temas a ser estudiados o resueltos. El término probablemente fue acuñado por Pitágoras (570 - 495 A.C). Los métodos filosóficos incluyen la reflexión, la discusión crítica, el argumento racional, y la presentación sistemática. La filosofía clásica tiene algunas preguntas como: ¿es posible saber algo y demostrarlo? ¿Qué es más real?

También plantean otras preguntas mucho más concretas, como: ¿hay una mejor forma de vivir? ¿Es mejor ser justo o injusto (en caso de que alguien se pueda librar de las consecuencias)? ¿Somos libres?

Historicamente, la "filosofía.enˆglobaba cualquier tipo de conocimiento. Desde los tiempos del antiguo griego Aristóteles hasta el siglo 19, la "filosofía natural"ha englobado la astronomía, la medicina, y la física. Por ejemplo, el libro de Newton `"Mathematical Principles of Natural Philosophy"` se convirtió después en un libro bajo la categoría de física. En el siglo 19, el crecimiento de universidades de investigación modernas provocó que la filosofía académica y otras disciplinas se profesionalizaran y especializaran. En la era moderna, algunas investigaciones que eran tradicionalmente parte de la filosofía se convirtieron en ramas independientes, incluyendo algunas como: la psicología, la sociología, la lingüística, y la economía.

Otras investigaciones estrechas al arte, ciencia, políticas u otras búsquedas siguen formando parte de la filosofía. Por ejemplo: ¿es la belleza objetiva o subjetiva? ¿Hay muchos métodos científicos o solo uno? ¿Es la política utópica un sueño alentador o una fantasía sin futuro? Algunas ramas de la filosofía académica incluyen la metafísica (çon lo que respecta a los fundamentos de la naturaleza, la realidad y el ser"), la epistemología (acerca de la "naturaleza y las bases del conocimiento [y]...sus límites y validez"), ética, estética, filosofía política, lógica y la filosofía de la ciencia.

Capítulo II

Instrucciones generales

- Tu proyecto debe estar escrito siguiendo la Norma. Si tienes archivos o funciones adicionales, estas están incluidas en la verificación de la Norma y tendrás un `0` si hay algún error de norma dentro.
- Tus funciones no deben terminar de forma inesperada (segfault, bus error, double free, etc) ni teener comportamientos indefinidos. Si esto pasa tu proyecto será considerado no funcional y recibirás un `0` durante la evaluación.
- Toda la memoria alocada en heap deberá liberarse adecuadamente cuando sea necesario. No se permitirán leaks de memoria.
- Si el subject lo requiere, deberás entregar un `Makefile` que compilará tus archivos fuente al output requerido con las flags `-Wall`, `-Werror` y `-Wextra`, por supuesto tu `Makefile` no debe hacer relink.
- Tu `Makefile` debe contener al menos las normas `$(NAME)`, `all`, `clean`, `fclean` y `re`.
- Para entregar los bonus de tu proyecto, deberás incluir una regla `bonus` en tu `Makefile`, en la que añadirás todos los headers, librerías o funciones que estén prohibidas en la parte principal del proyecto. Los bonus deben estar en archivos distintos `_bonus.{c/h}`. La parte obligatoria y los bonus se evalúan por separado.
- Si tu proyecto permite el uso de la `libft`, deberás copiar su fuente y sus `Makefile` asociados en un directorio `libft` con su correspondiente `Makefile`. El `Makefile` de tu proyecto debe compilar primero la librería utilizando su `Makefile`, y después compilar el proyecto.
- Te recomendamos crear programas de prueba para tu proyecto, aunque este trabajo **no será entregado ni evaluado**. Te dará la oportunidad de verificar que tu programa funciona correctamente durante tu evaluación y la de otros compañeros. Y sí, tienes permitido utilizar estas pruebas durante tu evaluación o la de otros compañeros.
- Entrega tu trabajo a tu repositorio `Git` asignado. Solo el trabajo de tu repositorio `Git` será evaluado. Si Deepthought evalúa tu trabajo, lo hará después de tus compañeros. Si se encuentra un error durante la evaluación de Deepthought, la evalaución terminará.

Capítulo III

Descripción general

Qué debes entender para completar correctamente el proyecto:

- Uno o más filósofos se sientan en una mesa redonda ya sea comiendo, pensando, o durmiendo. Mientras comen, no piensan o duermen. Mientras piensan, ni comen ni duermen. Por supuesto, mientras duermen, no comen o piensan.
- Los filósofos se sientan en una mesa redonda con un gran bol de espaguetis en el centro.
- Hay algunos tenedores en la mesa. Servir y comer espaguetis con un solo tenedor puede llegar a ser complejo, así que los filósofos comerán con dos tenedores. Uno en cada mano.
- Cada vez que un filósofo termine de comer, dejará los tenedores y empezará a dormir. Cuando terminen de dormir, empezarán a pensar. La simulación terminará tras la muerte de un filósofo.
- Todos los filósofos necesitan comer y nunca deben morirse de hambre.
- Los filósofos no se comunican entre ellos.
- Los filósofos no saben cuándo otro filósofo va a morir.
- No debería hacer falta decir que todos deben evitar morir.

Capítulo IV

Instrucciones generales

Debes escribir este programa para la parte obligatoria y otro para la parte bonus pero ambos seguirán el mismo conjunto de reglas básicas:

- Las variables globales están prohibidas.
- El programa debe aceptar los siguientes argumentos:
  - `number_of_philosophers`: es el número de filósofos, pero también el número de tenedores.
  - `time_to_die`: en **milisegundos**, si desde el principio de la simulación, o desde el principio de la última comida, un filósofo no empieza a comer... Morirá.
  - `time_to_eat`: en **milisegundos**, es el tiempo que tiene un filósofo para comer. Durante ese tiempo tendrá ambos tenedores ocupados.
  - `time_to_sleep`: en **milisegundos**, es el tiempo que el filósofo utilizará para dormir.
  - `number_of_times_each_philosopher_must_eat`: el argumento es opcional, si todos los filósofos comen al menos "number_of_times_each_philosopher_must_eat" la simulación se detendrá. Si no se especifica, la simulación se detendrá con la muerte de un filósofo.
- Cada filósofo tendrá asignado un número del 1 al "number_of_philosophers".
- El filósofo número 1 se sienta al lado del filósofo "number_of_philosophers". Todos los filósofos (N) se sientan entre otros dos filósofos (N - 1 y N + 1).

Sobre los logs de tu programa:

- Cualquier cambio de estado de un filósofo debe escribirse de la siguiente manera (cambia X por el número del filósofo y timestamp_in_ms por la marca temporal en milisegundos):
  - timestamp_in_ms X has taken a fork
  - timestamp_in_ms X is eating
  - timestamp_in_ms X is sleeping
  - timestamp_in_ms X is thinking
  - timestamp_in_ms X died
- El estado impreso no debe estar roto o alterado por el estado de otros filósofos.
- No puedes tener más de 10ms entre la muerte de un filósofo y el momento en el que imprime su muerte.
- De nuevo, los filósofos deben evitar morir.

Capítulo V

Parte obligatoria

Nombre de programa	`philo`
Archivos a entregar	`philo/`
Makefile	Sí
Argumentos	`number_of_philosophers time_to_die time_to_eat time_to_sleep [number_of_times_each_philosopher_must_eat]`
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	philosopher con hilos y mutex

Las reglas específicas para la parte obligatoria son:

- Cada filósofo debe ser un hilo.
- Un tenedor entre cada filósofo, de forma que si hay múltiples filósofos, tendrán un tenedor a la derecha y a la izquierda de cada uno.
- Para evitar que varios filósofos utilicen el mismo tenedor, deberás proteger el estado de cada uno con un mutex propio.

Capítulo VI

Bonus

Nombre de programa	`philo_bonus`
Archivos a entregar	`philo_bonus/`
Makefile	`Sí`
Argumentos	`number_of_philosophers time_to_die time_to_eat time_to_sleep [number_of_times_each_philosopher_must_eat]`
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	`philosopher con procesos y semáforos`

Para la parte bonus, el programa toma los mismos argumentos que antes y debe comportarse como se indica en la parte de reglas generales. 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 lo representa un semáforo.
- Cada filósofo debe ser un proceso y el proceso principal no debe ser un filósofo.