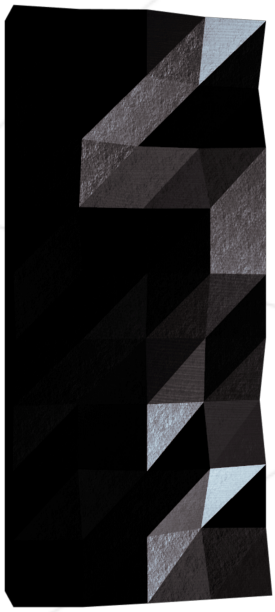

Philosophers

Nunca pensé que la filosofía llegara a ser tan letal

Resumen:
En este proyecto, aprenderás los principios básico de hilar un proceso.
Vas a aprender a como crear hilos y descubrirás los mutex.

Versión: 10

Índice general

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?

Históricamente, la “filosofía” englobaba 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 (“con 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 deberá estar escrito en C.
- 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 en cualquiera de ellos.
- Tus funciones no deben terminar de forma inesperada (segfault, bus error, double free, etc) ni tener comportamientos indefinidos. Si esto pasa tu proyecto será considerado no funcional y recibirás un `0` durante la evaluación.
- Toda la memoria asignada en el heap deberá liberarse adecuadamente cuando sea necesario. No se permitirán leaks de memoria.
- Si el enunciado lo requiere, deberás entregar un `Makefile` que compilará tus archivos fuente al output requerido con las flags `-Wall`, `-Werror` y `-Wextra` y 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 en tu repositorio `Git` asignado. Solo el trabajo de tu repositorio `Git` será evaluado. Si Deepthought evalúa tu trabajo, lo hará después de tus com-

pañeros. Si se encuentra un error durante la evaluación de Deepthought, esta habrá terminado.

Capítulo III

Descripción general

Aquí tienes una lista de cosas que deberías conocer si quieres superar este proyecto:

- Uno o más filósofos se sientan en una mesa redonda.
  En el centro de la mesa se encuentra un gran bol de espaguetis.
- Los filósofos solo pueden **comer**, **pensar**, o **dormir**.
  Mientras están comiendo, no pueden pensar ni dormir;
  Mientras están pensando, no pueden dormir ni comer;
  Y, por supuesto, mientras están durmiendo, no pueden comer ni pensar.
- También hay tenedores en la mesa, **tantos tenedores como filósofos**.
- Porque coger y comer espaguetis con un solo tenedor puede ser incomodo, los filósofos deben tomar el tenedor de la derecha y el de la izquierda, uno en cada mano.
- Cuándo un filósofo termine de comer, dejará los tenedores en la mesa e inmediatamente empezará a dormir. Una vez se despierte, empezará a pensar nuevamente. La simulación se detendrá cuando un filósofo muere por inanición.
- Todos los filósofos necesitan comer y nunca deben morir de hambre.
- Los filósofos no hablan entre ellos.
- Los filósofos no saben si otro filósofo va a morir.
- ¡No debería hacer falta decir que todos deben evitar morir!

Capítulo IV

Instrucciones generales

Deberás escribir un programa para la parte obligatoria y otro para la parte bonus (Solo si decides hacer la parte bonus). Ambas tienen que cumplir con las siguientes reglas:

- ¡Las variables globales están prohibidas!
- Tu(s) programa(s) debe(n) aceptar los siguientes argumentos:
  `number_of_philosophers` `time_to_die` `time_to_eat` `time_to_sleep`
  [`number_of_times_each_philosopher_must_eat`]
  - `number_of_philosophers`: es el número de filósofos, pero también el número de tenedores.
  - `time_to_die` (en milisegundos): si un filósofo no empieza a comer en `time_to_die` milisegundos desde el comienzo de su ultima comida o desde el principio de la simulación, este morirá.
  - `time_to_eat` (en milisegundos): es el tiempo que tiene un filósofo para comer. Durante ese tiempo, tendrá los tenedores ocupados.
  - `time_to_sleep` (en milisegundos): es el tiempo que tiene un filósofo para dormir.
  - `number_of_times_each_philosopher_must_eat` (argumento opcional): si todos los filósofos comen al menos `number_of_times_each_philosopher_must_eat` veces, 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 sentará al lado del filósofo número `number_of_philosophers`. Cualquier otro filósofo número `N` se sentarán entre el filósofo número `N` - 1 y el filósofo número `N` + 1.

Los logs de tu programa:

- Cualquier cambio de estado de un filósofo debe tener el siguiente formato:
  - `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`

  Reemplaza `timestamp_in_ms` *con la marca de tiempo actual en milisegundos y* `X` *con el numero del filósofo.*
- 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 imprimes su muerte.
- Te recuerdo, los filósofos deben evitar morir.

`Tu programa no debe tener ningún` **`data races.`**

Capítulo V

Parte obligatoria

Nombre de programa	`philo`
Archivos a entregar	`Makefile, *.h, *.c, en el directorio philo/`
Makefile	`NAME, all, clean, fclean, re`
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	`Philosophers con hilos y mutex`

Las reglas específicas para la parte obligatoria son:

- Cada filósofo debe ser un hilo.
- Hay un tenedor entre cada filósofo. por lo tanto, si hay varios filósofos, cada filósofo debe tener un tenedor a su izquierda y otro a su derecha. si solo hay un filósofo, solo habrá un tenedor en la mesa.
- Para prevenir que los filósofos dupliquen los tenedores, deberás proteger los estados de los tenedores con un mutex por cada uno de ellos.

Capítulo VI

Bonus

Nombre de programa	`philo_bonus`
Archivos a entregar	`Makefile, *.h, *.c, en el directorio philo_bonus/`
Makefile	`NAME, all, clean, fclean, re`
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	`Philosophers con procesos y semaforos`

En la parte bonus, el programa tendrá los mismos argumentos que la parte obligatoria. Y tendrá el mismo comportamiento como se indica en el capitulo de *Instrucciones generales*.

Las reglas específicas para la parte bonus son:

- Los tenedores están en el centro de la mesa.
- Los tenedores no tienen estados de memoria, pero el numero de disponibilidades está representados por un semaforo.
- Cada filósofo debe ser un proceso, y el proceso principal no debe ser un filósofo.

`Tus bonus serán evaluados exclusivamente si la parte obligatoria es EXCELENTE. Esto quiere decir, evidentemente, que debes completar la parte obligatoria, de principio a fin, y que tu gestión de errores debe ser impecable aunque el programa se utilice incorrectamente. De no ser así, esta parte será IGNORADA.`

Capítulo VII

Entrega y evaluación

Entrega tu proyecto en tu repositorio `Git` como de costumbre. Solo el trabajo en tu repositorio será evaluado durante la defensa. No dudes en verificar dos veces los nombres de tus archivos para segurarte que sean los correctos.

Directorio parte obligatoria : `philo/`

Directorio parte bonus : `philo_bonus/`