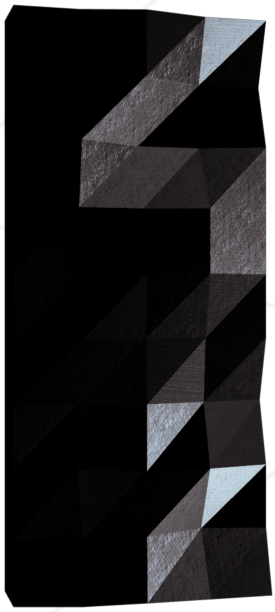

Philosophers

Nunca pensé que la filosofía llegara a ser tan letal

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

Versión: 13.0

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

- El proyecto deberá estar escrito en C.
- El proyecto debe estar escrito siguiendo la Norma. Si tienes archivos o funciones adicionales, estas deberán estar incluidas en la verificación de la Norma y tendrás un `0` si hay algún error de norma en cualquiera de ellos.
- Las funciones no deben terminar de forma inesperada (segfault, bus error, double free, etc), excepto en el caso de comportamientos indefinidos. Si esto sucede, el proyecto será considerado no funcional y recibirás un `0` durante la evaluación.
- Toda la memoria asignada en la pila (heap) deberá liberarse adecuadamente cuando sea necesario. No se permitirán leaks de memoria.
- Si el enunciado lo requiere, se deberá entregar un `Makefile` que compilará tus archivos fuente a la salida requerida con las flags `-Wall`, `-Werror` y `-Wextra`. También se deberá utilizar cc y, por supuesto, el `Makefile` no debe hacer relink.
- El `Makefile` entregado debe contener al menos las normas `$(NAME)`, `all`, `clean`, `fclean` y `re`.
- Para entregar los bonus del proyecto se deberá incluir una regla `bonus` en el `Makefile`, en la que se añadirán 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 el proyecto permite el uso de la `libft`, se deberá copiar su fuente y sus `Makefile` asociados en un directorio `libft` con su correspondiente `Makefile`. El `Makefile` del proyecto debe compilar primero la librería utilizando su `Makefile`, y después compilar el proyecto.
- Es recomendable crear programas de prueba para el proyecto, aunque este trabajo **no será entregado ni evaluado**. Esto ofrece la oportunidad de verificar que el programa funciona correctamente durante las evaluaciones. Y sí, está permitido utilizar estas pruebas durante cualquier evaluación.
- Entrega el trabajo en el repositorio `Git` asignado. Solo el trabajo de tu repositorio `Git` será evaluado. Si el proyecto tiene que ser evaluado por Deepthought, la evaluación se realizará después de las evaluaciones personales. Si, durante la evaluación de Deepthought, se encuentra un error se, se interrumpirá su evaluación.

Capítulo III

Instrucciones sobre la IA

Contexto

Durante tu proceso de aprendizaje, la IA puede ayudarte con muchas tareas diferentes. Tómate el tiempo necesario para explorar las diversas capacidades de las herramientas de IA y cómo pueden apoyarte con tu trabajo. Sin embargo, siempre debes abordarlas con precaución y evaluar de forma crítica los resultados. Ya sea código, documentación, ideas o explicaciones técnicas, nunca podrás saber con total certeza si tu pregunta está bien formulada o si el contenido generado es el adecuado. Las personas que te rodean son tu recurso más valioso para ayudarte a evitar errores y puntos ciegos.

Mensaje principal:

- Utiliza la IA para reducir las tareas repetitivas o tediosas.
- Desarrolla habilidades de prompting, ya sea para programacion o para otros temas, que beneficiarán tu futura carrera.
- Aprende cómo funcionan los sistemas de IA para anticipar de forma eficiente y evitar los riesgos comunes, sesgos y problemas éticos.
- Sigue trabajando con tus compañeros para desarrollar tanto habilidades técnicas como habilidades transversales.
- Utiliza únicamente contenido generado por IA que entiendas completamente y del cual puedas responsabilizarte.

Reglas para estudiantes:

- Debes tomarte el tiempo necesario para explorar las herramientas de IA y comprender cómo funcionan, para poder utilizarlas de manera ética y reducir los sesgos potenciales.
- Debes reflexionar sobre tu problema antes de dar instrucciones a la IA. Esto te ayuda a escribir preguntas, instrucciones o conjuntos de datos más claros, detalladas y relevantes utilizando un vocabulario preciso.

- Debes desarrollar el hábito de revisar, cuestionar y probar sistemáticamente cualquier contenido generado por la IA.
- Debes buscar siempre la revisión de otras personas, no te limites a confiar en tu propia validación.

Resultados de esta etapa:

- Desarrollar habilidades de prompting tanto generales como de ámbito especifico.
- Aumentar tu productividad con un uso eficaz de las herramientas de IA.
- Seguir fortaleciendo el pensamiento computacional, la resolución de problemas, la adaptabilidad y la colaboración.

Comentarios y ejemplos:

- Ten en cuenta que la IA puede no tener la respuesta correcta porque esa respuesta no esté ni siquiera en Internet. Además, si te da soluciones incorrectas, intenta no insistir y busca ayuda entre las personas que te rodean. Vas a ahorrarte tiempo y vas a sumar en compresión.
- Vas a enfretarte con frecuencia a situaciones (como exámenes o evaluaciones) donde debes demostrar una comprensión real. Prepárate, sigue construyendo tanto tus habilidades técnicas como transversales.
- Explicar tu razonamiento y debatir con otras personas suele revelar lagunas en tu comprensión de un concepto. Prioriza el aprendizaje entre pares.
- Lo normal es que la herramienta de IA que utilices no conozca tu contexto específico (a menos que se lo indiques), así que te dará respuestas genéricas. Si buscas información más adecuada y más precisa en relación a tu entorno cercano, confía en el resto de estudiantes.
- Donde la IA tiende a generar la respuesta más probable, el resto de estudiantes puede proporcionar perspectivas alternativas y matices valiosos. Confía en la comunidad de 42 como un punto de control de calidad.

✓ Buenas prácticas:

> Le pregunto a la IA: "¿Cómo pruebo una función de ordenación?"Me da algunas ideas. Las pruebo y reviso los resultados con otra persona. Refinamos el enfoque de manera conjunta.

✗ Mala práctica:

> Le pido a la IA que escriba una función completa, la copio y la pego en mi proyecto. Durante la evaluación entre pares, no puedo explicar qué hace ni por qué. Pierdo credibilidad. Suspendo mi proyecto.

✓ Buenas prácticas:

Utilizo la IA para ayudarme a diseñar un parser. Luego, reviso la lógica con otra persona. Encontramos dos errores y lo reescribimos juntos: mejor, más limpio y comprendiendo al 100

✗ Mala práctica:

Dejo que Copilot genere mi código para una parte clave de mi proyecto. Compila, pero no puedo explicar cómo maneja los pipes. Durante la evaluación, no puedo justificarlo y suspendo mi proyecto.

Capítulo IV

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 tomarán turnos para **comer**, **pensar** y **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.
- En la mesa habrá también **tantos tenedores como filósofos**.
- Como coger y comer espaguetis con un solo tenedor puede ser incómodo, 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 decirlo, pero ¡todos deben evitar morir!

Capítulo V

Instrucciones generales

Deberás escribir un programa para la parte obligatoria y otro para la parte bonus (Solo si decides hacer esta última). 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 que comenzó su ultima comida o desde el principio de la simulación, este morirá.
  - `time_to_eat` (en milisegundos): es el tiempo que tarda un filósofo en comer. Durante ese tiempo, tendrá los tenedores ocupados.
  - `time_to_sleep` (en milisegundos): es el tiempo que está un filósofo durmiendo.
  - `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 o no se detendrá.
- 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.

En relación a 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.*
- Los mensajes mostrados no deben estar incompletos ni superpuestos entre ellos.
- No pueden pasar más de 10ms entre la muerte de un filósofo y el momento en el que imprimes su muerte.
- Por si lo has olvidado, los filósofos deben evitar morir.

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

Capítulo VI

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 representado como un hilo independiente.
- 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 el estado de cada tenedor con un mutex.

Capítulo VII

Requisitos del Readme

Debe incluirse un archivo `README.md` en la raíz del repositorio Git. Su propósito es permitir que cualquier persona que no esté familiarizada con el proyecto (pares, personal, responsables de selección, etc.) pueda entender rápidamente de qué trata el proyecto, cómo ejecutarlo y dónde encontrar más información sobre el tema.
El `README.md` debe incluir, como mínimo:

- La primera línea debe estar en cursiva y decir: *Este proyecto ha sido creado como parte del currículo de 42 por <login1>[, <login2>[, <login3>[...]]].*
- Una sección de "**Descripción**"que presente claramente el proyecto, incluyendo su objetivo y una breve visión general.
- Una sección de "**Instrucciones**"que contenga cualquier información relevante sobre compilación, instalación y/o ejecución.
- Una sección de "**Recursos**"que enumere referencias clásicas relacionadas con el tema (documentación, artículos, tutoriales, etc.), así como una descripción del uso de IA, especificando para qué tareas y en qué partes del proyecto se ha utilizado.

➠ **Podrían requerirse secciones adicionales dependiendo del proyecto** (por ejemplo, ejemplos de uso, lista de características, decisiones técnicas, etc.).

Cualquier contenido extra requerida se listará explícitamente a continuación.

`Tu README.md debe estar escrito en inglés.`

Capítulo VIII

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 semáforos`

En la parte bonus, el programa tendrá los mismos argumentos que la parte obligatoria. Y deberá cumplir con los requisitos expuestos en el capítulo de *Instrucciones generales*.

Las reglas específicas para la parte bonus son:

- Todos loos tenedores están en el centro de la mesa.
- Los tenedores no tienen estados en memoria, pero el número de tenedores disponibles está representado por un semáforo.
- 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 IX

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 para la parte obligatoria: `philo/`

Directorio para la parte bonus: `philo_bonus/`

Durante la evaluación, es posible que se solicite una ligera **modificación del proyecto**. Esto puede consistir en ajustar ligeramente el comportamiento, modificar unas cuantas líneas de código o incorporar una característica fácil de implementar.

Puede que este paso **no sea necesario en todos los proyectos**, pero hay que tenerlo en cuenta si así se especifica en la hoja de evaluación.

Este paso sirve para verificar la comprensión real de una parte específica del proyecto. La modificación se puede realizar en cualquier entorno de desarrollo que se elija (por ejemplo, la configuración habitual), y debería ser factible en unos pocos minutos, a menos que se defina un plazo específico como parte de la evaluación.
Por ejemplo, se puede pedir hacer una pequeña actualización en una función o *script*, modificar lo que se vería en pantalla o ajustar una estructura de datos para almacenar nueva información, etc.

Los detalles (alcance, objetivo, etc.) se especificarán cada **hoja de evaluación** y pueden variar de una evaluación a otra para el mismo proyecto.