

# SCALE FOR PROJECT PHILOSOPHERS (/PROJECTS/42CURSUS-PHILOSOPHERS)

You should evaluate 1 student in this team



Git repository

git@vogosphere-v2.42madrid.com:vogosphere/intra-uuid-260dff36-3f54-44c



## Introduction

Por favor, respeta las siguientes directrices:


- Sé educado, cordial, respetuoso, y constructivo a lo largo del proceso de evaluación. El bienestar de la comunidad depende de ello.
- Identifica con la persona o grupo evaluados los posibles fallos de funcionalidad del trabajo. Tómate el tiempo de discutir y debatir los problemas identificados.
- Debes considerar la existencia de diferencias en cómo tus compañeros pueden haber entendido las instrucciones del proyecto y el alcance de las funcionalidades solicitadas.

## Guidelines

- Evalúa solo el trabajo entregado en el repositorio Git de la persona o grupo evaluado en la rama master y en el último commit.
- Comprueba dos veces que el repositorio de Git pertenece al estudiante o grupo evaluado. Asegúrate de que el trabajo entregado es el relativo al proyecto evaluado y que el comando "git clone" se utiliza en una carpeta vacía.
- Comprueba cuidadosamente que no se han utilizado alias maliciosos que puedan engañarte y hacerte evaluar trabajo ajeno al del repositorio oficial.
- Para evitar sorpresas, comprueba cuidadosamente que tanto el evaluador como el evaluado han revisado los scripts utilizados para facilitar la evaluación.
- Si el evaluador no ha completado este proyecto en particular todavía, es obligatorio que lea el subject por completo antes de empezar la evaluación.
- Utiliza las flags disponibles en esta evaluación para señalar un repositorio vacío, un programa disfuncional, con errores de Norma, trampas, etc. En estos casos, la evaluación termina y la nota final es 0 (o -42 en caso de trampa). Sin embargo, y a excepción de cuando hay trampas, se te recomienda seguir discutiendo el trabajo (aunque no esté finalizado) para identificar posibles fallos que puedan haber causado este error y evitar repetirlo en el futuro.
- Recuerda que, a lo largo de la evaluación, ningún segfault, ni terminaciones del programa prematuras, descontroladas, o inesperadas se tolerarán. En estos casos, la nota final es 0. Utiliza la flag apropiada. Nunca deberás editar archivos, salvo el de configuración si existe. Si quieres editar un archivo, tómate el tiempo de hacer explícitas las razones de por qué y verifica que el estudiante evaluado las entienda. Asegúrate de que ambos estéis de acuerdo con esto.
- Debes verificar la ausencia de data-races. Puedes utilizar cualquiera de las herramientas disponibles en el equipo, como valgrind con "--tool=helgrind" y "--tool=drd". En caso de encontrar cualquier data-race, la evaluación termina aquí.

- Debes verificar la ausencia de leaks de memoria. Toda la memoria alocada en el heap debe liberarse debidamente antes de que termine el programa. Puedes utilizar cualquier herramienta de detección disponible en el equipo, como leaks, valgrind o e\_fence. En caso de leaks de memoria, utiliza la flag apropiada.

## Attachments

 subject.pdf (<https://cdn.intra.42.fr/pdf/pdf/110425/es.subject.pdf>)

## Parte obligatoria

### Gestión de errores

Este proyecto se debe programar en C, siguiendo la Norma. Cualquier crash, comportamiento indefinido, leak de memoria, o fallo de Norma significa un 0 en el proyecto. En algún hardware lento, el proyecto puede no funcionar bien. Si algunas pruebas no funcionan en tu máquina, intenta discutirlo con calma y honestidad antes de invalidar el proyecto.

 Yes

 No

### Variables globales

Verifica si existe alguna variable global utilizada para gestionar los recursos compartidos entre filósofos. Si encuentras algún despropósito así, la evaluación termina aquí. Te animamos a que discutas el código, pero no puntúes el resto del proyecto.

 Yes

 No

### El código del filósofo

- Comprueba el código del filósofo cumple los siguientes requisitos y pide explicaciones de los mismos.
- Comprueba si hay un solo hilo por filósofo.
- Comprueba si hay un solo tenedor por filósofo.
- Comprueba si hay un mutex por tenedor y que se utiliza para comprobar el valor del tenedor y/o cambiarlo.
- Comprueba que el output nunca se mezcla.
- Comprueba cómo la muerte de un filósofo se comprueba. Verifica que hay un mutex para evitar que un filósofo muera y empiece a comer a la vez.

 Yes

 No

### Las pruebas del filósofo

- No pruebes con más de 200 filósofos.
- No pruebes con menos de 60 ms para time\_to\_die, time\_to\_eat o time\_to\_sleep.
- Prueba con 1 800 200 200, el filósofo no debe comer y debe morir.
- Prueba con 5 800 200 200. Nadie debería morir.
- Prueba con 5 800 200 200 7. Nadie debería morir y la simulación debería parar cuando todos los filósofos hayan comido como mínimo 7 veces cada uno.
- Prueba con 4 410 200 200. Nadie debería morir.
- Prueba con 4 310 200 100. Un filósofo debería morir.
- Prueba con 2 filósofos y verifica los distintos tiempos. Un retraso en la muerte de más de 10 ms es inaceptable.
- Prueba con los valores que elijas y verifica todos los requisitos. Comprueba que los filósofos mueren cuando toca, que no roban tenedores, etc.

 Yes

 No

## Bonus

### código del filósofo bonus

- Comprueba el código del filósofo bonus cumple los siguientes requisitos y pide explicaciones de los mismos.
- Comprueba que hay un solo proceso por filósofo y que el proceso principal no es un filósofo.
- Comprueba que no hay procesos huérfanos al acabar la ejecución del programa

- Comprueba si hay un solo semáforo que represente el número de tenedores.
- Valida si el output está protegido de múltiples accesos. Para evitar un output mezclado.
- Comprueba cómo se verifica la muerte de un filósofo y si hay un semáforo para evitar que un filósofo muera y empiece a comer a la vez.

 Yes

 No

### Los extras del filósofo

- No pruebes con más de 200 filósofos.
- No pruebes con menos de 60 ms para `time_to_die`, `time_to_eat` o `time_to_sleep`.
- Prueba con 5 800 200 200. Nadie debería morir.
- Prueba con 5 800 200 200 7. Nadie debería morir y la simulación debería parar cuando todos los filósofos hayan comido como mínimo 7 veces cada uno.
- Prueba con 4 410 200 200. Nadie debería morir.
- Prueba con 4 310 200 100. Un filósofo debería morir.
- Prueba con 2 filósofos y verifica los distintos tiempos (un retraso en la muerte de más de 10 ms es inaceptable).
- Prueba con tus valores para verificar todas los requisitos. Comprueba que los filósofos mueren cuando toca, que no roban tenedores, etc.

 Yes

 No

## Ratings


Don't forget to check the flag corresponding to the defense

 Ok

 Outstanding project

Empty work


 Incomplete work

 Invalid compilation

 Norme

 Cheat

 Crash

 Concerning situation

 Leaks

 Forbidden function

 Can't support / explain code

## Conclusion

Leave a comment on this evaluation

Finish evaluation

Declaration on the use of cookies  
(<https://profile.intra.42.fr/legal/terms/2>)

Privacy policy  
(<https://profile.intra.42.fr/legal/terms/5>)

General term of use of the site  
(<https://profile.intra.42.fr/legal/terms/6>)

Rules of procedure  
(<https://profile.intra.42.fr/legal/terms/4>)

Terms of use for video sur  
(<https://profile.intra.42.fr/legal/terms/3>)