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

You should evaluate 1 student in this team

★

Git repository

```
git@vogsphere-v2.42.fr:vogsphere/intra-uuid-efb4035b-bef0-4c56-80a6-
```

Introduction

Merci de respecter les règles suivantes:

- Restez polis, courtois, respectueux et constructifs pendant le processus d'évaluation. Le bien-être de la communauté repose là-dessus.

- Identifiez avec la personne évaluée ou le groupe évalué les éventuels dysfonctionnements de son travail. Prenez le temps d'en discuter et débattez des problèmes identifiés.

- Vous devez prendre en compte qu'il peut y avoir de légères différences d'interprétation entre les instructions du projet, son scope et ses fonctionnalités. Gardez un esprit ouvert et notez de la manière la plus honnête possible. La pédagogie n'est valide que si la peer-évaluation est faite sérieusement.

Guidelines

- Ne notez que ce qui est contenu dans le dépôt Git cloné de l'étudiant(e) ou du groupe.

- Vérifiez que le dépôt Git appartient bien à l'étudiant(e) ou au groupe, que le projet est bien celui attendu, et que "git clone" est utilisé dans un dossier vide.

- Vérifiez scrupuleusement qu'aucun alias n'a été utilisé pour vous tromper et assurez-vous que vous évaluez bien le rendu officiel.

- Afin d'éviter toute surprise, vérifiez avec l'étudiant(e) ou le groupe les potentiels scripts utilisés pour faciliter l'évaluation (par exemple, des scripts de tests ou d'automatisation).

- Si vous n'avez pas fait le projet que vous allez évaluer, vous devez lire le sujet en entier avant de commencer l'évaluation.

- Utilisez les flags disponibles pour signaler un rendu vide, un programme ne fonctionnant pas, une erreur de Norme, de la triche... Dans ces situations, l'évaluation est terminée et la note est 0, ou -42 en cas de triche. Cependant, à l'exception des cas de triche, vous êtes encouragé(e)s à continuer la discussion sur le travail rendu, même si ce dernier est incomplet. Ceci afin d'identifier les erreurs à ne pas reproduire dans le futur.

- Pendant toute la durée de l'évaluation, aucun segfault ou autre arrêt inattendu, prématuré ou incontrôlé ne sera toléré. Auquel cas, la note finale sera de 0. Utilisez le flag approprié.
Vous ne devriez jamais avoir à éditer un fichier hormis un fichier de configuration si existant. Dans le cas ou vous souhaitez modifier un fichier, vous devez expliciter clairement les raisons de l'édition et être en accord avec l'étudiant(e) évalué(e) avant de faire quoi que ce soit.

- Vous devez vérifier l'absence de data races.
Vous avez le droit d'utiliser tout outil disponible sur la machine tel que valgrind avec les options "--tool=helgrind" et "--tool=drd". En cas de data race, l'évaluation s'arrête ici.

- Vous devez aussi vérifier l'absence de fuites mémoire. Toute mémoire allouée sur le tas doit être libérée proprement avant la fin de l'exécution du programme.
Vous avez le droit d'utiliser tout outil disponible sur la machine tel que leaks, valgrind ou e_fence. En cas de fuites mémoire, cochez le flag approprié.

Attachments

subject.pdf (https://cdn.intra.42.fr/pdf/pdf/47563/fr.subject.pdf)

Partie obligatoire

Gestion d'erreur

Ce projet doit être codé en C en respectant la Norme. Tout crash, comportement indéterminé, fuite mémoire ou erreur de Norme signifie 0 au projet.

Yes No

Variables globales

Cherchez s'il y a une variable globale utilisée afin de gérer les ressources partagées entre les philosophes.
Si vous en trouvez, l'évaluation s'arrête ici. Vous pouvez continuer à regarder le code et échanger, mais ne le notez pas.

Yes No

Code de philo

- Vérifiez que le code de philo respecte les points suivants et demandez des explications :
- Assurez-vous qu'il y a un thread par philosophe.
- Assurez-vous qu'il y a bien une fourchette par philosophe.
- Assurez-vous qu'il y a un mutex par fourchette et qu'il est utilisé pour vérifier la valeur de la fourchette et/ou la changer.
- Vérifiez que l'output est protégé contre les accès multiples afin d'éviter un affichage mélangé.
- Vérifiez que la mort d'un philosophe est prise en compte et qu'il y a un mutex empêchant le philosophe de mourir et de commencer à manger exactement au même moment.

Yes No

Tests philo

- Ne testez pas avec plus de 200 philosphes.
- Ne testez pas avec time_to_die ou time_to_eat ou time_to_sleep set à une valeur inférieure à 60 ms.
- Testez avec 1 800 200 200. Le philosophe ne devrait pas manger et devrait mourir.
- Testez avec 5 800 200 200. Aucun philosophe ne devrait mourir.
- Testez avec 5 800 200 200 7. Aucun philosophe ne devrait mourir et la simulation devrait s'arrêter quand les philosophes ont mangé au moins 7 fois chacun.
- Testez avec 4 410 200 200. Aucun philosophe ne devrait mourir.
- Testez avec 4 310 200 100. Un philosophe devrait mourir.
- Testez avec 2 philosophes et vérifiez les timings (une mort prise en compte avec un retard de plus de 10 ms est inacceptable).
- Testez avec vos propres valeurs et vérifiez les règles. Vérifiez que les philosophes meurent au bon moment, qu'ils ne volent pas de fourchette, etc.

Yes No

Partie bonus

Code de philo_bonus

- Vérifiez que le code de philo_bonus respecte les points suivants et demandez des explications :
- Assurez-vous qu'il y a un processus par philosophe et que le processus initial attend les autres.
- Assurez-vous qu'il y a un unique sémaphore pour représenter le nombre de fouchettes.
- Vérifiez que l'output est protégé contre les accès multiples afin d'éviter un affichage mélangé.
- Regardez comment la mort d'un philosophe est vérifiée et assurez-vous qu'un sémaphore empêche le philosophe de mourir et de commencer à manger exactement au même moment.

Yes No

Tests philo_bonus

- Ne testez pas avec plus de 200 philosphes.
- Ne testez pas avec time_to_die ou time_to_eat ou time_to_sleep set à une valeur inférieure à 60 ms.
- Testez avec 5 800 200 200. Aucun philosophe ne devrait mourir.
- Testez avec 5 800 200 200 7. Aucun philosophe ne devrait mourir et la simulation devrait s'arrêter quand les philosophes ont mangé au moins 7 fois chacun.
- Testez avec 4 410 200 200. Aucun philosophe ne devrait mourir.
- Testez avec 4 310 200 100. Un philosophe devrait mourir.
- Testez avec 2 philosophes et vérifiez les timings (une mort prise en compte avec un retard de plus de 10 ms est inacceptable).
- Testez avec vos propres valeurs et vérifiez les règles. Vérifiez que les philosophes meurent au bon moment, qu'ils ne volent pas de fourchette, 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

Conclusion

Leave a comment on this evaluation

Finish evaluation