Introduction

Merci de respecter les règles suivantes :

- Restez poli, 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.

Attachments

subject.pdf

Tests préliminaires

Si un cas de triche est suspecté, la notation et l'évaluation prennent fin immédiatement. Pour le signaler, sélectionnez le flag "Cheat". Faites attention à l'utiliser avec calme, précaution et discernement.

Prérequis

Vérifiez les points suivants :

- Les conditions d'accès à la soutenance sont respectées : personne(s) évaluée(s) présente(s), un rendu non vide lié appartenant bien, etc...
- Si aucun travail n'a été soumis (ou mauvais fichiers, mauvais répertoire, ou noms de fichiers erronés), la note est de 0 et le processus d'évaluation prend fin.
- Pas de dépôt vide (= rien dans le dépôt Git).
- Aucune erreur de Norme.
- Tricherie (= -42).
- Pas d'erreur de compilation ni de Makefile qui re-link.

Si tous ces points sont validés, cochez "Oui" et continuez l'évaluation.
Sinon, utilisez le flag approprié à la fin de l'évaluation !

Yes | No

Instructions générales

Instructions générales

- Si un crash ou une erreur se produit de manière inattendue (erreur de segmentation, bus error, affichage anormal, etc.), utilisez le flag "Crash" !
- Le Makefile compile l'exécutable avec les options demandées.
- L'exécutable est nommé "pipex".
- Aucune fonction interdite n'est utilisée.

Yes | No

Partie obligatoire

La commande "./pipex file1 cmd1 cmd2 file2" doit se comporter comme cette commande : "< file1 cmd1 | cmd2 > fichier2"

Gestion des erreurs et des arguments

- Le programme prend 4 arguments, ni plus, ni moins (sauf pour la partie bonus) et uniquement dans le bon ordre.
- La gestion des erreurs est correcte : fichiers (in)existants, droits des fichiers, binaire de commande (in)existants, etc.

Si ces points sont respectés, cochez "Oui" et poursuivez l'évaluation.
Dans le cas contraire, l'évaluation s'arrête ici. Utilisez "Incomplete work" (rendu incomplet) ou tout autre flag approprié.

Yes | No

Le programme

Le programme fait ce qui est demandé dans le sujet sans afficher des informations/étapes supplémentaires par rapport à la commande shell.

Faites vos propres tests et comparez les résultats du programme avec ceux de la sortie de la commande originale. N'hésitez pas à regarder le sujet et les exemples donnés.

Si aucune erreur n'est détectée, cochez "Oui" et continuez. Dans le cas contraire, l'évaluation s'arrête ici.

Yes | No

Partie bonus

Les bonus ne seront examinés que si la partie obligatoire est excellente. Cela signifie que la partie obligatoire doit avoir été réalisée du début à la fin, avec une gestion d'erreur parfaite même en cas d'usage inattendu. Si tous les points obligatoires n'ont pas été attribués pendant cette soutenance, aucun point bonus ne sera comptabilisé.

Plusieurs pipes

Le programme gère l'utilisation de plusieurs pipes les uns après les autres.
Comme pour la partie obligatoire, testez avec les commandes shell afin de comparer avec la sortie du programme.

Si c'est bon seulement pour 2 pipes dans la même commande mais pas pour 3, ce bonus ne doit pas être compté.

Yes | No

<< et >> avec le paramètre here_doc

Le programme réplique l'utilisation de << et >>.

Testez plusieurs fois quelque chose comme :
"CMD << STOP_VALUE | CMD1 >> fichier1"

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