

(https://profile.intra.42.fr)

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

You should evaluate 2 students in this team



Git repository

git@vogsphere.42nice.fr:vogsphere/intra-uuid-2a69c0ae-369c-



## 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 débattrez 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, prémature 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 où 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 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 \(https://cdn.intra.42.fr/pdf/pdf/46025/fr.subject.pdf\)](https://cdn.intra.42.fr/pdf/pdf/46025/fr.subject.pdf)

## Partie obligatoire

---

### Compilation

- Utilisez "make -n" to pour vous assurer que le projet compile avec "-Wall -Wextra -Werror". Si ce n'est pas le cas, cochez le flag "invalid compilation".

- Le minishell compile sans aucune erreur. Si ce n'est pas le cas, cochez le flag.

- Le Makefile ne doit pas re-link. Si ce n'est pas le cas, cochez le flag.

 Yes

 No

### Commande simple et variables globales

- Exécutez une commande simple avec un PATH absolu tel que `/bin/ls` ou n'importe quelle autre commande sans option.
- Combien de variables globales y a-t-il ? Pourquoi ? Demandez à la personne évaluée de vous donner un exemple concret pour démontrer que leur usage est obligatoire et cohérent.
- Testez une commande vide.
- Testez seulement des espaces et des tabs.
- En cas de crash, cochez le flag "crash".
- Si quelque chose ne marche pas, cochez le flag "incomplete work".

☒ Yes☐ No

### Arguments et historique

- Exécutez une commande simple avec un PATH absolu tel que `/bin/ls` ou n'importe quelle autre commande, avec option mais sans " (single quotes) ni "" (double quotes).
- Répétez ce test plusieurs fois avec différentes commandes et différents arguments.
- En cas de crash, cochez le flag "crash".
- Si quelque chose ne marche pas, cochez le flag "incomplete work".

☒ Yes☐ No

### echo

- Lancez la commande echo avec et sans argument ou options, ou avec l'option -n.
- Répétez ce test plusieurs fois avec différents arguments.
- En cas de crash, cochez le flag "crash".
- Si quelque chose ne marche pas, cochez le flag "incomplete work".

☒ Yes☐ No

### exit

- Lancez la commande exit avec et sans arguments.
- Répétez ce test plusieurs fois avec différents arguments.
- N'oubliez pas de relancer le minishell.
- En cas de crash, cochez le flag "crash".
- Si quelque chose ne marche pas, cochez le flag "incomplete work".

☒ Yes☐ No

## Valeur de retour d'un processus

- Exécutez des commandes simple avec un chemin absolu tel que `/bin/l`s ou n'importe quelle autre commande avec des arguments mais sans " (single quotes) ni "" (double quotes), puis lancez `"echo $?"`. Vérifiez la valeur affichée. Vous pouvez le refaire dans bash et comparer.
- Répétez ce test plusieurs fois avec différentes commandes et différents arguments.
- Utilisez des commandes qui ne fonctionnent pas telles que `'/bin/l`s fichier nul'.
- Essayez des expressions telles que `$? + $?`
- En cas de crash, cochez le flag "crash".
- Si quelque chose ne marche pas, cochez le flag "incomplete work"

☒ Yes

☐ No

## Signaux

- ctrl-C dans un prompt vide devrait afficher une nouvelle ligne avec un nouveau prompt.
- ctrl-\ dans un prompt vide ne devrait rien faire.
- ctrl-D dans un prompt vide devrait quitter minishell. Ensuite, relancez-le.
- ctrl-C dans un prompt après avoir écrit des choses devrait afficher une nouvelle ligne avec un nouveau prompt.
- Également, le buffer devrait être vide. Appuyez sur "Entrée" afin de vous assurer que la ligne précédente a été exécutée.
- ctrl-D dans un prompt après avoir écrit des choses ne devrait rien faire.
- ctrl-\ dans un prompt après avoir écrit des choses ne devrait rien faire.
- Essayez ctrl-C après avoir lancé une commande bloquante, comme `cat` ou `grep` sans argument.
- Essayez ctrl-\ après avoir lancé une commande bloquante, comme `cat` ou `grep` sans argument.
- Essayez ctrl-D après avoir lancé une commande bloquante, comme `cat` ou `grep` sans argument.
- Répétez plusieurs fois en utilisant des commandes différentes.
- En cas de crash, cochez le flag "crash".
- Si quelque chose ne marche pas, cochez le flag "incomplete work"

☒ Yes

☐ No

## Double Quotes

- Exécutez une commande simple avec des arguments, mais cette fois utilisez des guillemets (rajoutez des ';' et des espaces entre les guillemets).
- Essayez une commande comme : `echo "cat lol.c | cat > lol.c"`

- N'essayez pas \$.
- En cas de crash, cochez le flag "crash".
- Si quelque chose ne marche pas, cochez le flag "incomplete work".

☒ Yes☐ No

---

### Single Quotes

- Exécutez des commandes avec des single quotes dans les arguments.
- Essayez des arguments vides.
- Faites des tests avec des variables d'environnement, des espaces, des pipes, des redirections entre les guillemets.
- echo '\$USER' doit afficher "\$USER"
- Rien ne devrait être interprété.

☒ Yes☐ No

---

### env

- Vérifiez qu'env vous affiche bien les variables d'environnement.

☒ Yes☐ No

---

### export

- Exportez des variables d'environnement, dont certaines pour remplacer les anciennes.
- Vérifiez le résultat avec env.

☒ Yes☐ No

---

### unset

- Exportez des variables d'environnement, dont certaines pour remplacer les anciennes.
- Utilisez unset pour en retirer.
- Vérifiez le résultat avec env.

☒ Yes☐ No

---

### cd

- Utilisez la commande cd pour vous déplacer dans l'arborescence et utilisez /bin/ls pour vérifier que vous êtes dans le bon répertoire.
- Répétez ce test plusieurs fois avec des cd qui fonctionnent et qui

ne fonctionnent pas.

- Essayez aussi '.' et '..' en arguments.

☒ Yes

☐ No

---

### **pwd**

- Utilisez la commande pwd, avec et sans argument.

- Répétez ce test plusieurs fois dans différents répertoires.

- Essayez '.' et '..' en arguments.

☒ Yes

☐ No

---

### **Chemin relatif**

- Exécutez des commandes en utilisant un chemin relatif.

- Répétez ce test plusieurs fois dans d'autres dossier avec un chemin relatif complexe (beaucoup de ..).

☒ Yes

☐ No

---

### **PATH d'environnement**

- Exécutez des commandes mais sans PATH (ls, wc, awk, etc...).

- Retirez le \$PATH et vérifiez si les commandes ne fonctionnent plus.

- Mettez plusieurs répertoires à PATH (directory1:directory2) et vérifiez qu'ils sont bien évalués de gauche à droite.

☒ Yes

☐ No

---

### **Redirection**

- Exécutez des commandes avec les redirections < et/ou >

- Répétez ce test plusieurs fois avec différentes commandes et différents arguments et, quelques fois, utilisez >> au lieu de >.

- Vérifiez si plusieurs instances de la même redirection échouent.

- Testez les redirections avec << (cela ne doit pas forcément mettre à jour l'historique).

☒ Yes

☐ No

---

### **Pipes**

- Exécutez des commandes avec des pipes telles que 'cat file | grep bla | more'

- Répétez plusieurs fois avec différentes commandes et différents arguments.

- Essayez des commandes qui échouent telles que 'ls fichiennul | grep bla | more'

- Mixez les pipes et les redirections.

☒ Yes☐ No

---

### Soyons fous ! Et l'historique

- Entrez une commande, puis ctrl-C, et appuyez sur "Entrée".  
Le buffer devrait être vide et il ne devrait plus rien avoir à exécuter.
- Peut-on naviguer dans l'historique avec Haut et Bas (profitez-en pour relancer des commandes) ?
- Exécutez des commandes qui ne fonctionnent pas telles que 'dskdskdksd' et vérifiez que tout fonctionne comme prévu.
- 'cat | cat | ls' doit fonctionner.
- Essayez des commandes vraiment, vraiment longues avec des tonnes d'arguments.
- Amusez-vous avec ce superbe minishell et profitez-en.

☒ Yes☐ No

---

### Variables d'environnement

- Exécutez echo avec des variables d'environnement (\$variable) en argument.
- Assurez-vous que \$ est interprété correctement.
- Vérifiez que les guillemets autour des \$variables fonctionnent correctement (comme dans bash).
- Si USER n'existe pas, définissez-la.
- Ainsi, echo "\$USER" devrait afficher la valeur de \$USER.

☒ 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.*

---

### And, Or

- Lancez des commandes avec &&, || et des parenthèses, et vérifiez que tout fonctionne comme dans bash.

☒ Yes☐ No

---

### Wildcard

- Utilisez des wildcards en argument dans le répertoire courant.

 Yes No



### Surprise ! Ou pas...

- Mettez une valeur à la variable USER.
- echo "\$USER" doit afficher la valeur de la variable USER.
- echo "\$USER" doit afficher "\$USER".

 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 Incomplete group Concerning situation Leaks Forbidden function

## Conclusion

Leave a comment on this evaluation

Finish evaluation

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

Terms of use for video surveillance (<https://signin.intra.42.fr/legal/terms/1>)

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

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

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

Legal notices (<https://signin.intra.42.fr/legal/terms/3>)