Compile 1

Utilisez « make -n » pour voir si la compilation utilise « -Wall -Wextra -Werror ».

Simple Command & global variables 2

Exécuter , /bin/ls, ou toute autre commande sans aucune option.
Exécuter /bin/ls sans options.

Combien de variables globales sont utilisées ?
au minimum $PATH est nécessaire pour trouver la commande. D'autres comme $PWD, $HOME, $OLDPWD peuvent être impliquées.

Pourquoi ?
$PATH permet de localiser /bin/ls, $PWD peut être utilisé par ls pour afficher le chemin actuel.

Demandez à l'élève évalué de vous donner un exemple concret de pourquoi cela semble obligatoire ou logique.
Si $PATH est vide, taper ls ne fonctionnera pas, car le shell ne saura pas où chercher.

Testez une commande vide.
Testez uniquement les espaces ou les tabulations.
Une commande vide ("") ne doit rien exécuter.
Un input contenant uniquement des espaces ou tabulations ne doit rien exécuter non plus.

Arguments 3
Exécuter, commande avec des arguments mais, sans guillemets ou doubles guillemets.

ls -l

echo hello world

grep word file.txt

cat file.txt

sans guillemets ou doubles guillemets.

grep word file.txt, cat file.txt (arguments sont bien séparés sans guillemets.)

Répétez l'opération plusieurs fois avec des arguments différents.

---
echo 4

Exécute la commande echo avec ou sans arguments, ou avec l'option -n.
(ne doit pas sauter de ligne)) (-n)

echo avec et sans arguments.

echo -n (ne doit pas ajouter de saut de ligne).

echo "hello world".

gestion des espaces multiples : echo "hello   world".

---

exit 5

Exécute la commande exit avec ou sans arguments.
exit

Répétez l'opération plusieurs fois avec des arguments différents.

exit 42 (devrait fermer le shell avec 42 comme code de retour)

exit hello doit renvoyer une erreur

Return value of a process 6

Exécuter /bin/ls, ou toute autre commande avec des arguments mais sans guillemets ni doubles guillemets.
(/bin/ls)
sans guillemets ni doubles guillemets. Exécutez ensuite echo $?
puis echo $? (devrait retourner 0).

Vérifiez la valeur imprimée. Vous pouvez faire la même chose en bash afin de comparer les résultats.
ls fichier_inexistant), puis echo $? (devrait être différent de 0).
Répétez l'opération , Essayez des commandes erronées comme '/bin/ls , fichier qui n'existe pas'
Essayez quelque chose comme expr $? + $?
expr $? + $?

Signals 7
ctrl-C dans une invite vide devrait afficher une nouvelle ligne avec une nouvelle invite.
ctrl-\ dans une invite vide ne doit rien faire.
ctrl-D dans une invite vide devrait quitter minishell --> RELAUNCH !
ctrl-C dans une invite après avoir écrit quelque chose devrait afficher une nouvelle ligne avec une nouvelle invite.
Le tampon doit être propre aussi. Appuyez sur « Enter » pour vous assurer que rien de la ligne précédente n'est exécuté.
ctrl-D dans une invite après avoir écrit des choses ne devrait rien faire.
ctrl-\ dans une invite après avoir écrit des choses ne devrait rien faire.
Essayez ctrl-C après avoir lancé une commande bloquante comme cat sans arguments ou grep « quelque chose ».
Essayez ctrl-\ après avoir lancé une commande bloquante comme cat sans arguments ou grep « quelque chose ».

Essayez ctrl-D après avoir exécuté une commande de blocage comme cat sans arguments ou grep « quelque chose ».
Répétez l'opération plusieurs fois en utilisant des commandes différentes.

Double Quotes 8

Exécutez une commande simple avec des arguments et, cette fois, utilisez également des guillemets doubles (vous devriez essayer d'inclure des
les espaces blancs).

echo "hello world" (doit afficher hello world).
echo "cat file | grep word" (doit afficher la ligne sans exécuter).
$USER dans echo "$USER" (doit afficher la valeur de $USER).

Essayez une commande comme : echo « cat lol.c | cat > lol.c »
Essayez tout sauf $.

Single Quotes 9

Exécute les commandes dont les arguments sont des guillemets simples.
echo '$USER' doit afficher $USER littéralement.

Essayez les arguments vides.
echo 'hello world'
hello world
echo 'ls | grep test'
ls | grep test
echo 'ls > output.txt'
ls > output.txt
echo '$USER'

Essayez les variables d'environnement, les espaces, les tuyaux, les redirections entre guillemets simples.
echo '$USER' doit imprimer « $USER ».
Rien ne doit être interprété.

---
env 10

Vérifier si env affiche les variables d'environnement actuelles.

env

resultat : ex
HOME=/home/user
PATH=/usr/bin:/bin:/usr/sbin:/sbin
USER=user
SHELL=/bin/bash
PWD=/home/user

---

export 11

Exporte les variables d'environnement, en crée de nouvelles et remplace les anciennes.
Vérifiez le résultat avec env.

export VAR=test puis echo $VAR doit afficher test.
Vérifier avec env.

---

unset 12

Exporte les variables d'environnement, en crée de nouvelles et remplace les anciennes.
Utilisez unset pour en supprimer certaines.
Vérifiez le résultat avec env.

unset VAR, puis echo $VAR doit ne rien afficher.
Vérifier avec env.

---

cd 13

Utilisez la commande cd pour déplacer le répertoire de travail et vérifiez que vous êtes dans le bon répertoire avec /bin/ls
Répétez l'opération plusieurs fois avec des cd qui fonctionnent et d'autres qui ne fonctionnent pas.
Essayez également '.' et '...' comme arguments.

cd
Tester cd /tmp, cd .., cd -.
Tester cd . et cd ... (ce dernier ne devrait pas fonctionner).

---

pwd 14

Utilisez la commande pwd.
pwd
Répétez l'opération plusieurs fois dans différents répertoires.
pwd pwd pwd

Relative Path 15

Exécuter les commandes en utilisant cette fois un chemin relatif.
Répéter plusieurs fois dans différents répertoires avec un chemin relatif complexe (beaucoup de ..).

/home/user/Documents/,

cd ../,
tu seras dans /home/user/.

Environment path 16

Exécutez les commandes, mais cette fois sans chemin d'accès (ls, wc, awk, etc.).
ls
wc
awk

Décomposer le $PATH et s'assurer que les commandes ne fonctionnent plus.

echo $PATH (/usr/local/bin:/usr/bin:/bin:/usr/sbin:/sbin)

ou trouver ls (which ls) = /bin/ls ou /usr/bin/ls

Définir $PATH à une valeur de plusieurs répertoires
(répertoire1:répertoire2) et s'assurer que les répertoires sont vérifiés dans l'ordre de gauche à droite.

export PATH=/tmp:/usr/bin:/bin

dans l'ordre, de gauche à droite.

echo "echo FAKE LS" > /tmp/ls

chmod +x /tmp/ls

ls

FAKE LS (Cela montre que le shell exécute bien /tmp/ls avant /usr/bin/ls, car /tmp est en premier dans $PATH.)

Redirection 17

Exécuter des commandes avec des redirections < et/ou >
Répéter plusieurs fois avec des commandes et des arguments différents et parfois remplacer > par >>.
Vérifier si les tentatives multiples des mêmes redirections échouent.
Tester << la redirection (il n'est pas nécessaire de mettre à jour l'historique).

Tester ls > output.txt, cat < input.txt, echo "test" >> output.txt.
Vérifier que > remplace et >> ajoute.
Tester << (heredoc).

Pipes 18

Exécuter des commandes avec des tuyaux comme « cat file | grep bla | more ».
Répétez l'opération plusieurs fois avec des commandes et des arguments différents.

cat file | grep word | wc -l.
ls fileinexistant | grep word (invalide)

Essayez des commandes erronées comme « ls filethatdoesntexist | grep bla | more
Essayer de mélanger les tuyaux et les redirections.

Go Crazy and history 19

Tapez une ligne de commande, puis utilisez ctrl-C et appuyez sur « Enter ».
La mémoire tampon doit être propre et il ne doit
rien à exécuter.

Vérifier ctrl-C sur une ligne (ne doit rien exécuter)

Pouvons-nous naviguer dans l'historique en utilisant les touches Haut et Bas ?
Peut-on réessayer une commande ?

Tester navigation avec flèches haut/bas.

Exécuter des commandes qui ne devraient pas fonctionner comme 'dsbksdgbksdghsd'.
Assurez-vous que le minishell ne se bloque pas et n'affiche pas d'erreur.
erreur.

Tester une commande invalide (dfsgfdsg)

La commande 'cat | cat | ls' devrait se comporter de manière « normale ».
Essayez d'exécuter une longue commande avec une tonne d'arguments.

Tester cat | cat | ls (doit fonctionner)

Environment variables 20

Exécutez echo avec certaines variables d'environnement ($variable) comme arguments.
Vérifiez que $ est interprété comme une variable d'environnement.
Vérifiez que les guillemets doubles interpolent $.
Vérifiez que USER existe. Dans le cas contraire, définissez-le.
echo « $USER » devrait afficher la valeur de la variable USER.

Tester echo $USER (doit afficher l’utilisateur).
Tester echo "$USER" (doit fonctionner pareil).
Tester echo '$USER' (doit afficher $USER littéralement).

