Introduction

Nous vous demandons pour le bon déroulement de cette évaluation de respecter les règles suivantes :

- Restez courtois, polis, respectueux et constructifs en toutes situations lors de cet échange. Le lien de confiance entre la communauté 42 et vous en dépend.
- Mettez en évidence auprès de la personne (ou du groupe) notée les dysfonctionnements éventuels du travail rendu, et prenez le temps d'en discuter et d'en débattre.
- Acceptez qu'il puisse y avoir parfois des différences d'interprétation sur les demandes du sujet ou l'étendue des fonctionnalités. Restez ouvert d'esprit face à la vision de l'autre (a-t-il ou elle raison ou tort ?), et notez le plus honnêtement possible. La pédagogie de 42 n'a de sens que si la peer-évaluation est faite sérieusement.

Guidelines

- Vous ne devez évaluer que ce qui se trouve sur le depôt GiT de rendu de l'étudiant(e) ou du groupe.
- Prenez soin de vérifier que le depôt GiT est bien celui correspondant a l'étudiant(e) ou au groupe, et au projet.
- Verifiez méticuleusement qu'aucun alias malicieux n'a été utilisé pour vous induire en erreur et vous faire évaluer autre chose que le contenu du dépot officiel.
- Tout script sensé faciliter l'évaluation fourni par l'un des deux partis doit être rigoureusement vérifié par l'autre parti pour éviter des mauvaises surprises.
- Si l'étudiant(e) correcteur/correctrice n'a pas encore fait ce projet, il est obligatoire pour cet(te) etudiant(e) de lire le sujet en entier avant de commencer cette soutenance.
- Utilisez les flags disponibles sur ce barème pour signaler un rendu vide, non fonctionnel, une faute de norme, un cas de triche, etc. Dans ce cas, l'évaluation est terminée et la note finale est 0 (ou -42 dans le cas special de la triche). Toutefois, hors cas de triche, vous etes encouragés a continuer d'échanger autour du travail éffectué (ou non éffectué justement) pour identifier les problemes ayant entrainé cette situation et les éviter pour le prochain rendu.

Partie obligatoire

Rappel : si à un moment ou un autre, le programme ne réagit pas correctement (bus error, segfault, etc..), ou bien si vous détectez une fuite mémoire, la soutenance est terminée et la note est 0. Pensez à utiliser les flags correspondants quand cela est nécessaire. Cette consigne est active d'un bout à l'autre de la soutenance.

Fichier auteur

Vérifiez que le fichier "auteur" est bien présent à la racine du dépot et formaté tel que demandé dans le sujet. Dans le cas contraire, la soutenance est terminée et la note est 0.

Fuites mémoire

Pendant toute la durée de la soutenance, gardez un oeil sur la quantité de mémoire utilisée par le minishell (à l'aide de top par exemple). Cette quantité doit rester à peu près fixe commande après commande. Dans le cas contraire, il y a au moins une fuite mémoire, la note du projet est 0.

Fork et execve

"fork" et "execve" sont au coeur de la base d'un shell minimaliste, tel que le minishell. Si vous constatez que ces deux fonctions ne sont jamais appelées dans le code source du programme, c'est qu'il y a un problème de compréhension du sujet. La soutenance est terminé et la note est 0. En respectant la liste des fonctions autorisées, il n'y a pas d'autre solution.

Effectuez les 4 tests suivants. Si l'un au moins de ces tests échoue, alors toute la section est échouée, passez à la suivante :

- Lancez le minishell, puis lancez la commande "$> foo". La commande doit échouer avec un message d'erreur adapté et rendre le prompt.
- Lancez la commande "$> /bin/ls". Le binaire ls doit s'exécuter correctement, puis le prompt doit réapparaitre.
- Lancez la commande "$> /bin/ls -laF". Le binaire doit s'exécuter correctement avec les options -l, -a et -F, puis le prompt doit réapparaitre,
- Lancez la commande "$> /bin/ls -l -a -F". Le binaire doit s'exécuter correctement avec les options -l, -a et -F, puis le prompt doit réapparaitre

Les builtins

Nous allons évaluer dans cette section l'implémentation des builtins "exit", "echo" et "cd". Même préhistorique, un shell doit proposer certaines fonctionnalites.

Effectuez les 6 tests suivants. Si l'un au moins de ces tests échoue, alors toute la section est échouée, passez à la suivante :

- Lancez le minishell, puis lancez la commande "$> exit". Le programme doit se terminer proprement et rendre la main au shell parent. Relancez le minishell.
- Lancez une commande telle que "$> echo It works". Le message doit s'afficher correctement.
- Lancez une commande telle que "$> cd /absolute/path/of/your/choice", puis lancez la commande "$> /bin/pwd". La commande /bin/pwd doit confirmer que le dossier courant a bien été changé.
- Lancez une commande telle que "$> cd relative/path/of/your/choice", puis lancez la commande "$> /bin/pwd". La commande /bin/pwd doit confirmer que le dossier courant a bien été changé.
- Lancez la commande "$> cd", puis lancez la commande "$> /bin/pwd". La commande /bin/pwd doit confirmer que le dossier courant est bien le home de l'utilisateur.
- Lancez la commande "$> cd -", puis lancez la commande "$> /bin/pwd". La commande /bin/pwd doit confirmer que le dossier courant est bien le dossier relative/path/of/your/choice précédent.

Gestion de l'environnement

Nous allons évaluer dans cette section l'implémentation des builtins "env", "setenv" et "unsetenv".

Effectuez les 6 tests suivants. Si l'un au moins de ces tests échoue, alors toute la section est échouée, passez à la suivante :

- Lancez la commande "$> env". Les variables d'environnement doivent s'afficher sous la forme clef=valeur.
- Lancez une commande telle que "$> setenv FOO bar" ou "$> setenv FOO=bar" selon la syntaxe choisie pour setenv. Lancez ensuite la commande "$> env". L'environnement doit afficher une variable FOO avec pour valeur bar.
- Lancez la commande "$> /usr/bin/env". Le minishell doit transmettre le bon environnement aux binaires qu'il lance. Le binaire /usr/bin/env doit donc afficher l'environnement incluant une variable FOO avec pour valeur bar en s'exécutant.
- Lancez la commande "$> unsetenv FOO". Lancez ensuite la commande "$> env". L'environnement ne doit plus afficher une variable FOO avec pour valeur bar.
- Lancez à nouveau la commande "$> unsetenv FOO". Lancez ensuite à nouveau la commande "$> env". L'environnement ne doit pas avoir été modifé.
- Lancez à nouveau la commande "$> /usr/bin/env". Le binaire /usr/bin/env doit ne doit plus afficher l'environnement incluant une variable FOO avec pour valeur bar en s'exécutant.

Gestion du PATH

Nous allons évaluer dans cette section la gestion de la variable d'environnement PATH par votre shell.

Effectuez les 6 tests suivants. Si l'un au moins de ces tests échoue, alors toute la section est échouée, passez à la suivante :

- Lancez la commande "$> unsetenv PATH", puis la commande "$> setenv PATH "/bin:/usr/bin"" ou "$> setenv "PATH=/bin:/usr/bin"" selon la syntaxe choisie. Lancez ensuite la commande "$> ls". Le binaire /bin/ls doit s'exécuter correctement.
- Lancez la commande "$> emacs". Le binaire /usr/bin/emacs doit s'exécuter correctement.
- Lancez la commande "$> unsetenv PATH", puis lancez la commande "$> ls". L'exécution doit échouer.
- Lancez à présent la commande "$> emacs". L'exécution doit échouer également.
- Lancez la commande "$> /bin/ls". Le binaire /bin/ls doit s'exécuter correctement.
- Lancez la commande "$> /usr/bin/emacs". Le binaire /usr/bin/emacs doit s'exécuter correctement.

Gestion de la ligne de commande

Nous allons évaluer dans cette section la gestion de la ligne de commande. Effectuez les 4 tests suivants. Si l'un au moins de ces tests échoue, alors toute la section est échouée, passez à la suivante :

- Lancez une commande vide "$> ". Le shell ne doit rien faire et réafficher le prompt.
- Lancez une commande composée uniquement d'un seul espace "$> ". Le shell ne doit rien faire et réafficher le prompt.
- Lancez une commande composée uniquement d'espaces et de tabulations. Le shell ne doit rien faire et réafficher le prompt.

- Lancez une commande avec plusieurs espaces et tabulations avant le nom du binaire, entre chaque argument passé au binaire, et après le dernier argument. Ces espaces et tabulations inutiles ne doivent pas perturber l'éxécution de la commande.

Pipe

Dans cette section nous allons tester la gestion du pipe.

Effectuez les 4 tests suivants. Si l'un au moins de ces tests échoue, alors toute la section est échouée, passez à la suivante :

- Executez la commande suivante "$> ls | cat -e". Le shell doit afficher le contenu du dossier courant avec un '$' a la fin de chaque ligne.
- Executez la commande suivante "$> ls | sort | cat -e". Le shell doit afficher le contenu du dossier courant trié avec un '$' a la fin de chaque ligne.
- Executez la commande suivante "$> base64 /dev/urandom | head -c 1000 | grep 42 | wc -l | sed -e 's/1/Yes/g' -e 's/0/No/g'". Le shell doit afficher "Yes" si la string "42" est présente dans la chaine de caracteres aléatoire, sinon il doit afficher "No".

Redirections simple

Dans cette section nous allons tester les redirections.

Effectuez les 3 tests suivants. Si l'un au moins de ces tests échoue, alors toute la section est échouée, passez à la suivante :

- Executez la commande suivante "$> echo "Testing redirections," > /tmp/test.txt" et vérifiez que le fichier /tmp/test.txt contient la string "Testing redirections,".
- Executez la commande suivante "$> echo "with multiple lines" >> /tmp/test.txt" et vérifiez que le fichier /tmp/test.txt contient la string "Testing redirections,". et "with multiple lines" sur la ligne suivante.
- Executez la commande suivante "$> wc -c < /tmp/test.txt" et vérifiez que le retour est bien '42'.

Logical operators

Dans cette section nous allons évaluer l'implementation des opérateurs logique.

Effectuez les 3 tests suivants. Si l'un au moins de ces tests échoue, alors toute la section est échouée, passez à la suivante :

- Executez la commande suivante "$> ls -l && ls" et verifiez que la commande ls s'est bien executée 2 fois de manière differentes.
- Executez la commande suivante "$> ls something || ls" et vérifiez que la premiere execution est un echec et que la seconde fonctionne bien.
- Executez la commande suivante "$> ls || ls something " et verifiez que seulement le premier 'ls' est executé.

Séparateur
Dans cette section nous allons évaluer la gestion de plusieurs commandes se suivants séparé par un ";".

Effectuez le test suivant. Si l'un au moins de ces tests échoue, alors toute la section est échouée, passez à la suivante :

- Executez la commande suivante "$> ls -1; touch test_file; ls -1". Les 2 'ls' doivent s'executer à la difference que le second afficher un nouveau fichier 'test_file' en plus.

A lil bit of everything
Dans cette section nous allons évaluer le pipe, les redirections et le séparateur dans la meme commande.

Effectuez le test suivant. Si l'un au moins de ces tests échoue, alors toute la section est échouée, passez à la suivante :

- Executez la commande suivante "$> mkdir test ; cd test ; ls -a ; ls | cat | wc -c > fifi ; cat fifi" La sortie devrait être la suivante:

. ..

5

Partie optionnelle
Rappel : vous ne devez évaluer la partie optionnelle que si la partie obligatoire est PARFAITE. Il n'y a aucun intérêt à développer des fonctionnalités exotiques sur un shell qui n'assure pas de manière parfaite les fonctionnalités de base !

Inhibiteurs
Dans cette section nous allons tester la presence et le bon fonctionnement des inhibiteurs """ (double quote), "'" (simple quote) et "\" (backslash).

Effectuez les 3 tests suivants. Si l'un au moins de ces tests échoue, alors toute la section est échouée, passez à la suivante :

- Executez la commande suivante '$> echo "' vérifiez que le shell attend bien une fermeture des doubles quotes. Tappez du texte et finissez par une double quote. Le texte devrait s'imprimer comme un 'echo' normal.
- Executez la commande suivante "$> echo '" vérifiez que le shell attend bien une fermeture de la simple quote. Tappez du texte et finissez par une double quote. Le texte devrait s'imprimer comme un 'echo' normal.
- Executez la commande suivante "$> ls \" assurez vous bien que le shell attend le reste de la commande. Tappez n'importe quel flag que vous souhaitez tester. Le commande doit etre executée avec les flag que vous avez choisi.

Heredoc

Dans cette section nous allons tester l'implemetation du heredoc.

Effectuez le test suivant. Si l'un au moins de ces tests échoue, alors toute la section est échouée, passez à la suivante :

- Executez la commande suivante "$> cd /tmp; sort << EOF | cat -e > sorted_poem ; sed -e 's/Roses/Turnips/' < sorted_poem > better_poem; cd -; echo "I prefer turnips anyway" >> /tmp/better_poem; cat /tmp/better_poem" et tappez ce poeme suivant sans les double quotes mais avec les retours à la ligne:

"""

Roses are red

Violets are blue

All my base are belong to you

I love you

"""

La sortie devrait etre celle-ci:

"""

All my bases are belong to you$

I love you$

Turnips are red$

Violets are blue$

I prefer turnips anyway

"""

Aggrégation des file descriptors de sortie

Dans cette section nous allons évaluer l'existence et le fonctionnement de l'aggrégation des file descriptors de sortie.

Effectuez les 3 tests suivants. Si l'un au moins de ces tests échoue, alors toute la section est échouée, passez à la suivante :

- Executez la commande suivante "$> rm nosuchfile 2>&-" en vérifiant qu'aucun fichier nommé "nosuchfile" est présent dans le dossier courant. Le message d'erreur ne doit pas s'afficher.

- Executez la commande suivante "$> rm nosuchfile 2>&1 | cat -e" en vérifiant qu'aucun fichier nommé "nosuchfile" est présent dans le dossier courant. Le sortie standard devrait afficher: "rm: nosuchfile: No such file or directory$"
- Executez la commande suivante "echo "No dollar character" 1>&2 | cat -e". Le sortie standard devrait afficher: "No dollar character".

Edition de ligne

Dans cette section nous allons tester l'edition de ligne.

Effectuez les 5 tests suivants. Si l'un au moins de ces tests échoue, alors toute la section est échouée, passez à la suivante :

- Il doit etre possible de deplacer le curseur a gauche ou a droite dans la ligne de commande avec les fleches du clavier.
- Il doit etre possible de modifier la line la ou le curseur se trouve.
- Il doit etre possible de se deplacer au debut et a la fin d'une ligne en utilisant les touche home et end (ou d'autres).
- Il doit être possible de naviguer entre les commandes entrée précédemment en utilisant les touches haut et bas du clavier.
- Il doit etre possible de copier/coller une partie ou toute la ligne de commande avec un raccourcie clavier.

Back quotes

Dans cette section nous allons évaluer l'implémentation des back quotes.

Effectuez le test suivant. Si l'un au moins de ces tests échoue, alors toute la section est échouée, passez à la suivante ::

- Executez la commande suivante "$> echo `ls -l`" et verifiez que 'echo' imprime bien le resultat de 'ls -l' et non la commande elle-meme.

Globing

Testez la presence et le bon fonctionnement du globing ("*", "?", "[]", "{}", etc). Regardez dans les sources l'implémentation du globbing, la fonction glob(3) ne doit pas être utilisée.

Sous-shell

Dans cette section nous allons évaluer l'implémentation du sous-shell.

Testez la presence et le bon fonctionnement des sous shells ("()").

Variables locales

Dans cette section nous allons tester l'implémentation des variables locales.

Testez la presence et le bon fonctionnement des variables locales et des builtins 'unset' et 'export'.

Historique
Dans cette section nous alons tester l'implementation de l'historique.

Testez la presence et le bon fonctionnement de l'historique des commandes et des builtins history et "!" avec toutes leurs options si elles en ont.

Manipulation des fichiers
Dans cette section nous allons tester l'implémentation de la manipulation de fichier.

Testez l'existence et le bon fonctionnement des descripteurs de fichiers et de la builtin 'read' avec toutes ses options.

Auto-completion
Dans cette section nous allons tester l'implémentation de l'auto-completion.

Testez en ecrivant le debut d'une commande ou un path suivie de 'TAB' si la commande est auto-completée.

Job control
Dans cette section nous allons tester l'implémentation du job control. Testez la presence et le bon fonctionnement du Job control et des builtins 'job', 'fg', 'bg', et de l'operateur "&".

Bonus
Shell script
Testez la présence et le bon fonctionnement du shell script.

D'autres fonctionnalitées
Si le 42sh a des fonctionnalitées supplémentaires, comptabilisez-les ici. Vous pouvez comptabiliser jusqu'à 5 fonctionnalitées bonus. Les bonus doivent être 100% fonctionnels et ne pas mettre en cause la stabilité du shell.