Travail pratique #1 – IFT-2245

Abdelhakim Qbaich
Rémi Langevin

February 11, 2018

Problèmes rencontrés

Gérer les assignements de variables multiples

Étant donné que nous voulions éviter de gérer les erreurs dûes à des caractères invalides lors de l'utilisation des variables, nous avons utiliser des expressions régulières afin de "whitelist" les caractères valides. Toutefois, cela a rendu plus hardu la gestion de variables de 'shell'. Ainsi la commande suivante n'est pas valide:

```
PATH=miaw HOME=wouf echo $HOME $PATH
```

Toutefois, il est possible d'assigner multiples variables d'environnement comme suit :

```
COCO=lapin MIAW=RonRonRon
```

Boucle "for"

Plusieurs idées nous sont venues en tête afin d'implémenter la boucle "for". Nous avons d'abord essayer d'avoir des triples pointeurs. Toutefois, cela s'avère affreux et complexe à gérer. Nous avons ensuite essayer d'avoir un struct contenant des pointeurs vers le début de chaque partie d'une boucle "for" (i.e. variable d'itération, valeurs et le corps de la boucle). Cela s'est avéré bien mieux et nous avons peaufiner le tout autour de cette méthode. Ensuite, avec quelque peu de récursion, nous avons réussi à gérer les for imbriqués.

"cd -"

Pour le moment, nous ne supportons pas la commande "cd -", pour revenir au répertoire précédent (temporellement). Toutefois, il ne s'agit que de bien gérer et mettre à jour les variables PWD et OLDPWD.

Surprises

Le travail a été plus ardu qu'initialement prévu. Le manque de temps n'a pas aidé. La manipulation de string n'est pas si aisée que ça, comparativement à des langages plus haut niveau.

Choix

Variables d'environnement non-existante

Lorsque que le shell a une requête pour retrouver une variable d'environnement qui n'existe pas, nous retournons une string vide plutôt que de l'ignorer.

```
echo $HOME $MIAW $HOME
```

Gestion du CTRL+C

Nous avons choisi de ne pas catch les SIGINT générés par les touches CTRL+C. Ainsi, le shell est interrompu plutôt que le "child process" seulement.

Options rejetées

Yacc/Lex Bison/Flex

L'idée originale avec ce projet était d'utiliser Yacc/Lex ou Bison/Flex pour générer une grammaire et son parseur, ce qui simplifierait énormément la tâche, considérant que le parsing des expressions et un des éléments les plus compliqués, surtout lorsqu'on parle de récursitivité dans les boucles.

Cependant, par manque de temps, l'étude de la syntaxe byzantine de Yacc/Lex n'était pas une si bonne option, surtout pour un shell relativement de base.

En conséquent, nous avons utiliser strsep(3). Un choix plutôt typique, mais qui représente une légère amélioration par rapport à l'utilisation de strtok(3).

Commandes built-in à l'intérieur d'une boucle "for"

Nous avons rejeté l'option de pouvoir exécuter des commandes tel que "cd" dans une boucle "for".

Outre cela, nous n'avons pas particulièrement rejeté d'options, à l'exception d'options avancées (e.g. completion/suggestion, pipe, etc.)

Ajouts

Bindings emacs et historique

Avec la libraire readline, nous avons pu facilement avoir les bindings emacs et gérer l'historique de la session. De plus, cela permet de déplacer le curseur sur la ligne de commande.

Easter Egg

Essayer les commandes suivantes:

```
twado                              # pour souligner le twadoversary
sudo rm -rf --no-preserve-root /   # pour du bon temps
```