Programmation Système — Projet

1 Objectifs

Il s'agit d'écrire un *shell* permettant d'interpréter des expressions et lançant des processus pour les exécuter. La grammaire suivante décrit la forme minimale des expressions qui doivent être traitées (ε désigne le mot vide). Il est possible d'ajouter d'autres constructions.

expression $\longrightarrow$	ε
|	`commande`
|	*expression* `;` *expression*
|	*expression* `||` *expression*
|	*expression* `&&` *expression*
|	`(`*expression*`)`
|	*expression* `&`
|	*expression* `|` *expression*
|	*expression* `>desc fichier`
|	*expression* `<desc fichier`
|	*expression* `>>desc fichier`

La signification des différents symboles est la même qu'avec les shells habituels. Une commande pourra être soit interne, soit externe, et peut évidemment comporter des arguments.

Les commandes internes suivantes devront en particulier être interprétées, avec leur signification habituelle : `alias`, `unalias`, `cd`, `dirs`, `popd`, `pushd`, `echo`, `exec`, `exit`, `history`, `kill`, `printenv`, `pwd`, `source`, `suspend`, `umask`, `times`, `wait`.

Ensuite, on réalisera au choix l'une de ses extensions suivantes :

1. Ajout du *job control*. Il s'agit d'ajouter les fonctionnalités pour
   — lancer des processus en arrière plan, par `commande &`.
   — stopper par `Ctrl-Z` le (groupe des) processus en premier plan,
   — envoyer à ce groupe des signaux générés au terminal (comme `SIGINT` et `SIGQUIT`).
   — ajouter les commandes internes `bg`, `fg`, `jobs`, `disown`, en utilisant le caractère `%` pour désigner un numéro de job.
2. Manipulation de l'environnement, en ajoutant la commande interne `export`, et complétion
   — des noms de fichiers. Le shell devra pouvoir interpréter les caractères spéciaux courant, comme `*`, `?`, `\`...
   — des variables d'environnement lorsqu'elles sont précédées du caractère `$` (on ne demande cependant pas d'écrire un mécanisme général de complétion).

2 Comment démarrer ?

Un embryon du projet vous est fourni. Vous pouvez vous concentrer sur les fichiers `Shell.h` `Shell.c`. En particulier le programme `Shell.c` est, en l'état, capable d'analyser les lignes de commande qui lui sont soumises. Pour réaliser cette analyse ce programme utilise une fonction d'*analyse syntaxique* qui renvoie l'arbre syntaxique associé à la ligne de commande [1]. C'est un arbre binaire qui décrit comment sont combinées les commandes contenues dans la ligne de commande donnée. Par exemple, une ligne de commande comme `ls -al | grep lol` sera décrite par un arbre dont la racine sera de type `PIPE`, cette racine ayant pour fils gauche une feuille de type `SIMPLE` contenant la commande simple `ls -al` et pour fils droit une autre feuille de type `SIMPLE` décrivant `grep lol`. On trouvera dans le fichier `Shell.c` une description de la structure de données utilisée.

Votre travail consiste à exploiter cet arbre syntaxique afin d'exécuter la ligne de commande entrée par l'utilisateur. Pour ce faire, il vous est demandé de modifier les fichiers fournis pour les adapter à vos développements.

3 Modalités

Équipe de projet. Le projet est à réaliser par équipes de trois étudiants. Les étudiants ne participant pas au contrôle continu peuvent contribuer au projet d'une équipe mais ne seront pas notés. L'équipe utilisera un dépôt svn abrité au CREMI pour le développement du projet. Un accès à ce dépôt devra être communiqué aux enseignants [2] dès sa création. N'hésitez pas à nous [3] contacter en cas de difficulté avant qu'il ne soit trop tard.

Soutenances. Les soutenances auront lieu le lundi 8 décembre. En plus d'une démonstration automatisée à l'aide d'un script, chaque étudiant présentera en soutenance la partie du travail qu'il a réalisé. Les enseignants pourront observer le dépôt pour mesurer la contribution de chacun. Un doodle sera organisé pour définir le planning des soutenances.

Documents à produire. Un document pdf regroupant les sources du projet ainsi qu'un rapport (5 à 15 pages) détaillant les choix effectués et les résultats [4] obtenus seront à transmettre aux enseignants le *dimanche 23 novembre* minuit.

Instructions à suivre pour remettre les documents. Suite au doodle un numéro de session vous sera assigné. Mettre dans une archive tar à la fois le rapport et le code. Le nom de l'archive aura le format suivant :

```
tar czf x-login1-login2-login3-CLEF-SECRETE.tar.gz rapport.pdf code/
```

où x est le numéro de votre session de soutenance, login(1,2,3) sont les logins des membres du trinôme, CLEF-SECRETE une suite de caractères de votre choix. Utiliser ensuite la commande suivante pour vérifier qu'il y a bien tout ce qu'il faut :

```
tar tzf x-login1-login2-login3-CLEF-SECRETE.tar.gz
```

S'assurer de donner le droit en lecture pour tous sans pour autant donner le droit en écriture :

```
chmod a+r x-login1-login2-login3-CLEF-SECRETE.tar.gz
```

1. NB. cette fonction retourne `NULL` en cas de ligne syntaxiquement incorrecte.
2. username : auesnard, aguermou, ahugo, pwacreni
3. abdou.guermouche@labri.fr, andra.hugo@inria.fr, aurelien.esnard@labri.fr, pierre-andre.wacrenier@labri.fr
4. En particulier on listera les commandes qui marchent réellement, c'est à dire dans tous les cas.

```
chmod go-w x-login1-login2-login3-CLEF-SECRETE.tar.gz
```

Déplacer enfin l'archive dans le répertoire /net/cremi/pwacreni/SHELL/. Les droits de ce répertoire sont `drwx----wt`, donc normalement : vous pouvez écrire dedans (ouf!), vous ne pouvez pas supprimer les TPs des autres (droit 't'), vous ne pouvez pas lire le contenu du répertoire (donc souvenez-vous bien de votre clé secrète...), `pwacreni` a tous les droits...