Université de Bordeaux — Programmation Système
Licence Informatique — Mini projet

Programmation d'un Interprète Shell

travail individuel à rendre sous moodle

Présentation

Un embryon de shell vous est fourni dans le répertoire `fork-exec`. Le programme `Shell.c` est, en l'état, capable d'analyser les lignes de commande qui lui sont soumises et d'afficher le résultat de son analyse. 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. Voici la définition de la structure de données utilisée :

```
  typedef enum expr_t {
   ET_EMPTY,          // Commande vide
   ET_SIMPLE,         // Commande simple
   ET_SEQUENCE,       // Séquence (;)
   ET_SEQUENCE_AND,   // Séquence conditionnelle (&&)
   ET_SEQUENCE_OR,    // Séquence conditionnelle (||)
   ET_BG,             // Tâche en arrière-plan (&)
   ET_PIPE,           // Tube (|)
   ET_REDIRECT,       // Redirection
} expr_t;

typedef enum redirection_t {
   REDIR_IN,          // Entrée (< ou <&)
   REDIR_OUT,         // Sortie (> ou >&)
   REDIR_APP,         // Sortie, mode ajout (>>)
} redirection_t;

typedef struct RedirectInfo {
   redirection_t type;
   int fd;            // -1 si &> ou &>>
   bool toOtherFd;    // si <& ou >&
   char * fileName;
} RedirectInfo;
```

```
typedef struct Expression Expression;
struct Expression {
   expr_t type;
   Expression * left;
   Expression * right;
   union {
      struct {
         char * * argv;
         int argc;
      };
      ArgsList argsList;
      RedirectInfo redirect;
   };
};
```

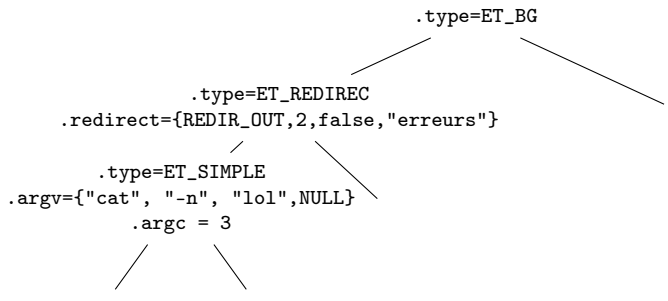

FIGURE 1 – Arbre associé à la commande `cat -n lol 2> erreur &`

1. NB. cette fonction retourne `NULL` en cas de ligne syntaxiquement incorrecte.

Compilez le programme et testez-le en entrant la commande `./Shell`. Dans l'état, le programme se contente d'afficher l'arbre associé à la commande. Par exemple la commande `cat -n lol 2> erreur &` provoque l'affichage suivant :

```
mini_shell(0):  cat -n lol 2> erreur &
 +-->BG
 |   +-->REDIRECT OUT [2] file [erreur]
 |   |   +-->SIMPLE [cat][-n][lol]
sorry, this shell is not yet implemented
mini_shell(1):
```

Les entiers `0` et `1`, apparaissant entre parenthèses dans le prompt, correspondent à la valeur de l'*exit status* de la dernière commande. Cette valeur est conservée dans la variable globale `shellStatus`. On notera que cette valeur n'est pas modifiée par l'exécution d'une commande vide.
Le fichier `Shell.c` comporte une description des structures de données employées. On pourra également consulter la fonction `printExpr()` pour voir comment explorer récursivement la structure de donnée associée à une expression.

Objectifs et plan de travail

Il s'agit d'implémenter dans le fichier `Evaluation.c` les fonctionnalités suivantes :

1. Exécution et redirection de commandes simples. Plan de travail suggéré :
   — exécuter une commande externe simple (telle que `ls -al`) ;
   — effectuer une redirection de la sortie standard (eg. `ls -al > output`), de l'entrée standard (eg. `cat -n < input`) ou de la sortie d'erreur standard (eg. `ls toto 2> erreurs`) ;
   — effectuer les redirections multiples de l'entrée et des sorties standard (eg. `cat -n < input > output`) ;
   — mettre en place le mode prolongation (`>>`) ;
   — mettre en place la redirection simultanée des sorties standard (`&>`).
2. Enchainement de commandes et utilisation de pipelines. Plan de travail suggéré :
   — évaluer une séquence de commandes (telle que `ls -al ; cat -n lol`) puis avec les opérateurs `&&` et `||` ;
   — évaluer des pipelines de deux commandes (tel que `ls -al | cat -n`) ;
   — évaluer des pipelines de plusieurs commandes.
3. Traitement des commandes en arrière plan. Il s'agit de récupérer dès que possible les zombies en réagissant au signal `SIGCHLD` [2]. Plan de travail suggéré :
   — mettre en arrière plan une commande (telle que `xeyes &`) ;
   — mettre en place l'élimination des zombies en utilisant le traitement des signaux.

Bonus : les étudiants motivés pourront enrichir leur travail en ajoutant la redirection vers un descripteur (`ls 2>&1 > autre`), des commandes internes (echo,cd,source,...), le traitement des variables d'environnement ou encore des traitements plus ambitieux comme le traitement des jobs ou la gestion du terminal. Ces extensions seront évaluées par votre enseignant de TD lors d'une démonstration / soutenance individuelle et quelques points pourront être ajoutés à la moyenne *moodle* de l'étudiant.

2. Indice : l'appel à la fonction `setpgid()` est à envisager.