Projet minishell

LAURENT Thomas
STEFANIAK Adam

Master 1 informatique 2017

Présentation de notre minishell.

1 Modules, Compilation et exécution

1.1 Modules

Le projet est décomposé en plusieurs modules:

lib/memoirePartager.c .h Fichier utilisé pour gérer une mémoire partagée sous la forme clé=valeur (clé=valeur), l'algorithme de gestion de la mémoire utilisé est la subdivision, l'implémentation consiste à jouer avec deux listes:
La première est une chaîne contenant tous les ensembles clés=valeurs ayant comme indice de premier caractère une puissance de 2.
La seconde est un tableau d'entier qui pour chaque entier non négatif est un indice vers chaque ensemble clé=valeur de la première chaîne.
Pour imprimer chaque ensemble clé=valeur de la première chaîne, on itère chaque entier du second tableau puis on affiche la chaîne à la position <adresse de la première chaîne> + <indice courant du tableau>.

lib/mesJobs.c .h Module ayant pour utilité de s'occuper de toute la partie des commandes 'myjob', 'myfg', 'mybg', une mémoire partagé (memoireParteger.h) est utilisée pour pouvoir stocker tous les processus mis en arrière plan.
Les fonctions liées aux signaux SIGTSTP, SIGCONT sont implémentées dans ce fichier.

lib/ouvrirRepertoire.c .h Ce module permet de travailler avec les fichiers.

lib/parseurDeCommande.c .h Ce module transforme la commande brute en liste de commandes, il vérifie si l'utilisateur ne marque pas synthaxiquement n'importe quoi et effectue les diverses transformations , par exemple s'll voit une expression régulière. Avant de traiter la suite de la commande brute, l'algorithme va exécuter l'expression régulière, si l'algorithme tombe sur un mot ayant comme premier caractère un $, il va directement chercher la valeur de la variable dans les segments de mémoire partagée.

lib/redirection.c .h Lorsque la commande qui va être exécuté contient un ou plusieurs caractères de redirection (2>> , <, ...), ce dernier appelle les fonctions de ce module pour changer la table des descripteurs du processus courant.

lib/status.c .h Enfin un petit module, il gère la commande 'status'.

lib/utilitiesRegex.c .h Insère dans la liste des commandes tous les répertoires ou fichiers qui match avec la regex donné. Après recherche via man pour savoir comment grep travaille avec le PATHERN passé en paramètre, j'ai trouvé la fonction glob.

lib/utilitesString.c .h Quelques fonctions utilitaires pour faciliter la manipulation des chaînes de caractères.

lib/CONST_mytinyshell.h Des constantes de préprocesseur qui déterminent si une suite de caractères est un opérateur logique, de redirection, de variable, ...

src/quivontbien.h Ce fichier contient tous les includes nécessaires au fonctionnement de chaque module du projet, il est suivit par des constantes de préprocesseur pour les codes d'erreurs, les erreurs, le test du fork etc.

src/myls.c .h Le programme externe demandé (-help est disponible).

src/myps.c .h Le programme externe demandé comparable au ps aux.

mytinyshell.c .h fait le lien entre tous les modules et implémente l'exécution des processus.

1.2 Compilation et exécution

Il faut se placer dans le répertoire /minishell. Vous êtes au bonne endroit si vous voyez un fichier nommé src et un autre nommé bin.

```
$ cd minishell/
$ ls
lib/  Makefile  mytinyshell.c  mytinyshell.h  src/
```

Voici la ligne de commande pour pouvoir compiler notre projet:

```
$ make
```

Puis celle-ci pour le lancer:

```
$ ./mysh
```

2 Tests unitaires

Lors de toutes les séquences de tests, on utilisera le caractère en début de ligne pour montrer dans quel type de shell nous nous trouvons:

> signifie que l'on écrit une commande dans ./mysh

>> signifie que l'ont écrit dans un ./mysh lancé depuis un autre ./mysh

$ signifie que l'on écrit sur le shell de l'os

$$ on écrit sur un autre shell de l'os

```
$ ./mysh
> ./mysh
>> exit
> exit
$ sh
$$ exit
$ exit
```

2.1 Séquencement inconditionnel

```
> echo a;echo b; echo c ;echo d ; echo e
a
b
c
d
e

> ;
Erreur
```

2.2 Séquencement conditionnel

```
> echo coucou&&echo salut && echo bonsoir&&
 echo annyeong &&echo hello
coucou
salut
bonsoir
annyeong
hello

> echo coucou || ls
coucou

> echo coucou || echo hello && echo annyeong
coucou
annyeong

>echo coucou && echo annyeong || echo hello
coucou
annyeong

>echop hello && echo bonjour; echo annyeong
annyeong
```

2.3 Expressions régulières

Les regex ont été testé dans le répertoire minishell/ Si la chaîne en cours d'analyse contient au moins un caractère utilisé dans les regex (et bien-sur n'ayant pas un anti slash à la position i-1) celle-ci sera envoyé dans le module regex.

```
> echo ???*/*      fonctionne
> echo ???/        fonctionne
> echo *.[^cho]    affiche *.[^cho]
> echo *.[ho]      fonctionne
> echo [A-M]*      affiche [A-M]*
> echo [A-MM]*     fonctionne
> echo [a-A]       affiche [a-A]
> echo [a-]        affiche [a-]
> echo []          affiche []
> echo m*y??*.c    fonctionne
> echo *?          fonctionne
> echo *           fonctionne
> echo ?*/*/*.h    fonctionne
> echo \*          affiche *
```

2.4 Changement de répertoire

```
> pwd
/home/.../Bureau/projetDeSec/minishell
> cd
> pwd
/home
> cd /etc
> pwd
/etc
> cd /home/???/Bureau
> pwd
/home/.../Bureau
```

2.5 Sortie du shell avec exit

```
> gedit &
 [0] 20045
> exit
$ ps x | grep gedit
20045 pts/2 Sl 0:00 gedit &
$ kill -9 20045
$ gedit (jobs=[0], pid=20045) terminer avec status -1
```

2.6 Sortir du shell avec CTRL+C

```
> gedit
> <ctrl+c>
> <ctrl+c>
Voulez vous sortir du shell ? (y/n):
abcdefghijklm n opqrstuvwxyz
> <ctrl+c>
Voulez vous sortir du shell ? (y/n):
y
$ ./mysh
> gedit &
 [0] 6063
> <ctrl+c>
Voulez vous sortir du shell ? (y/n):
y
$ ps x | grep gedit
```

2.7 Code de retour d'un processus

```
$ ./mysh
> status
Aucun processus n'a etait lance
> echo annyeong
annyeong
> status
echo termine avec comme code de retour 0
> gedit
$$ <ouverture d'un autre shell>
$$ ps x
$$ kill -9 <pid du processus gedit>
> status
gedit termine anormalement
> pwwd
Le programme [pwwd] n'existe pas
> status
pwwd termine avec comme code de retour 127
```

2.8 Redirections

Il y a quelque fois des erreurs de segmentation, mais très rarement.

```
> ls tt.txt
ls: impossible d'acceder a 'tt.txt': Aucun fichier
 ou dossier de ce type
> echo /etc/* > tt.txt
> more tt.txt
/etc/emacs /etc/X11 ......
> ls  >> tt.txt
more tt.txt
/etc/emacs /etc/X11 ... lib src ...
> wc tt.txt
 45 618 5631 tt.txt
> nl < tt.txt
<il affiche les 45 lignes>
> ls > tt.*
> cat tt.*
lib src ...
> rm tt.txt
```

2.9 Les Jobs

CTRL+Z Pour que le ctrl+z ne soit pas ignoré, il faut que la shm contenant le pid du processus actuellement en exécution en premier plan ne soit pas égal à 0, si cette variable contient autre choses que 0 alors il existe un programme actuellement en exécution en premier plan et le SIGTSTP se propagera vers ce processus. (la commande 'shmpid' donne le pid en premier plan, utile lors de mes tests donc je l'ai laissé).

myjobs Affiche tous les jobs en arrière plan et tous les jobs stoppés, s'il y a un job dont le pid n'existe plus alors l'ensemble job=pid est supprimé de la zone de mémoire partagée.

mybg Il faut entrer un numéro de job avec la commande. La fonction appellée par mybg réveille le processus avec SIGCONT. Si on appelle un processus qui est déjà en arrière plan, l'erreur sera affichée et aucune action ne sera effectuée sur ce processus.

myfg Il faut entrer un numéro de job avec la commande. La fonction appellée par myfg réveille le processus avec SIGCONT, retire le job=pid de la zone de mémoire partagée puis place en attente le processus 'main', tant que le processus du pid indiqué n'est pas mort, le 'main' ne reprendra pas la main. Il y a quelques problèmes si on utilise ces commandes avec plusieurs processus.

2.10 Variables

set La commande set donne sur une zone de mémoire partagée. La zone de mémoire partagée a comme clé, le répertoire '/bin/echo' et le pid du processus shell.

setenv La commande setenv donne sur une zone de mémoire partagée. cette zone est commune àtous les procesuss ayant la clef '/bin/echo' et la valeur 1.

unset et unsetenv Contrairement à l'énoncé, il ne faut pas placer de '$' avant le nom de variable pour qu'elle soit supprimer.

taille maximale Toutes les zones de mémoire partagée (crée via la librairie *memoirePartager.h*) ont une taille maximale donnée dans le *.h* ($TAILLE\_MEMOIRE\_PARTAGER\_DEFAUT$) valant 2048 caractères par défaut.
Toutes les commandes ne pouvant pas être insérer, résultera d'une erreur et un abandon de l'insertion.

```
> set
> setenv
SESSION=ubuntu
SHELL=/bin/bash
> setenv SHELL=mytinyshell
> setenv
SESSION=ubuntu
SHELL=mytinyshell
> set a=ls
> set b=$a
> set
a=ls
b=ls
> unsetenv a
Element non trouve
> unset SHELL
Element non trouve
> unset a
> set
b=ls
> ./mysh
>> set
>> setenv
SESSION=ubuntu
SHELL=mytinyshell
>> set b=5
>> setenv b=6
>> exit
> set
b=ls
> setenv
SESSION=ubuntu
SHELL=mytinyshell
b=6
```

2.11 Les tubes

Les tubes sont aussi implémentés et fonctionnent très bien pour les exemples suivants:

```
> ls | sort -r
...
> ls -al | sort -r | grep my | wc | wc -c
24
> echo coucou && ls -a | wc -c && echo fini
coucou
     21      21     218
fini
> echo coucou && ls -a | wccccc && fini
coucou
Erreur wccccc n'existe pas
> ls ???/*.[^co] | wc
     12      12     221
```

2.12 myls et myps

Ils se lancent comme suivant 'myls' 'myps'.

Précisions sur la commande myps:
Pour la commande myps, les données suivantes n'ont pas été incluses:
%MEM et tty.
Une fonction de récupération des données %CPU a cependant été faite.
Cette fonction consiste en la récupération des données utime et stime dans le fichier /proc/[pid]/stat ainsi que d'un temps total d'utilisation CPU.
Ensuite une seconde récupération de ces données est faite aprés une seconde d'attente. Le soucis est le fait qu'il y a une seconde d'attente pour chaque processus, ce qui nuit a l'ergonomie du programme.
Le code est présent dans le fichier myps.c mais l'appel vers cette fonction n'est pas faite.
Il se peut (rarement) qu'une segmentation se lance.

2.13 Conclusion

Pourcentage du travail réalisé:
LAURENT Thomas: 50%
STEFANIAK Adam: 50%

Blagounette (in english !):
A man is sunbathing on a nude beach.
To prevent a sunburn, he covers his most important organ with a hat.
A woman passes by and notices the hat.
She says, "Sir, a real gentleman always lifts his hat in front of a lady."
The man replies, "Ma'am if you were a real lady, the hat would've lifted itself."