UFR IM²AG

UNIVERSITÉ Grenoble Alpes

mini-shell.

BOUKRIS Walid
SOW Ousmane

1. Introduction:

Le but de ce projet et de faire un mini-shell, qui permet d'exécuter les commandes internes et externes, en prenant en compte la gestion des pipes ”|” des entrées/ sorties “</>” et le background “&” sans oublier les “;”.

Pour commencer on a essayé de bien comprendre les 2 premiers exercices de TP4-5, et le code de la page 43 de cours et suivre le même principe.

On a mit un journal “Journal.txt” contenant tous les avancement de projet durant ces 2 semaines, on vous invite à jeter un coup d'oeil!

On a consacré de temps pour bien commenter le code et pour que ça soit claire et bien simplifier.

2. Utilisations et tests:

Pour vous facilitez la correction on a mis en place un Makefile qui nous permet de compiler, et créer un fichier exécutable nommé “shell”.

Et aussi un script shell nommé “Exec.sh” qui fait appel au make et lance l’executable! Donc pour tester il suffit de tapez “./Exec.sh” sur le terminal!

Dans ce mini shell il faut mettre des espaces entre chaque commandes. On n’a pas essayé d'améliorer notre lecteur commandes pour ce concentrer sur les autres fonctionnalités!

Compilation du projet:

- par cible Make : make shell ou make

Exécution du projet : Commande :

- ./Exec.sh ou ./shell s’il existe!

3. Structures et Méthodes utilisées:

Structures readcmd:

Voir fichier “readcmd.h” qui est bien commenté! ce qui est important a savoir c’est que cette structure contient un char** seq qui contient toutes les commandes séparées par des “;”. Exemple d’utilisation:

la commande tapez : ls -al > ls ; cat < fic1 ; who | cat

seq[0] vaut : ls -al > ls

seq[1] vaut : cat < fic1

seq[2] vaut : who | cat

Méthodes utilisées:

Nom	Données	Fonction
Lire_commande	char* cmd, char** com	Rempli le tableau com et met chaque commande dans une case!
nbPipes	char** com	Renvoie le nombre de pipe existant.
readcmd	char** com	Renvoie un pointeur sur la structure cmdline. Avec tableau seq bien rempli
n_pipe	struct cmdline * s,char **com	Exécute des commandes simple avec et sans pipe.

4. Commandes internes:

Parmis les commandes internes on a eu à réaliser entre autre :

- La commande cd:
  Si on tapes cd sans paramètre on se déplace dans le répertoire HOME, en utilisant la méthode “chdir(getenv("HOME"));”
  Si on tapes cd avec un paramètre on se déplace dans le répertoire donnée en paramètre avec la même méthode “chdir”!
- La commande exit
  On ferme le programme avec “exit(-1);”
- La commande getenv
  Affiche la valeur d'environnement correspondant au paramètre donnée.
- La commande setenv
  Sans paramètre nous affiche toutes les variables d'environnement, avec leurs valeurs
  Avec paramètre permet de créer une nouvelle variable d'environnement avec sa valeur.

5. Gestions des Pipes:

Au début On a réalisé une méthode nommé “one_pipe” qui permet d'exécuter une ligne de commande contenant un seul pipe, Puis on a généralisé ce principe dans une méthode nommé “n_pipe” qui permet d'exécuter une ligne de commande contenant un nombre quelconque de pipe (>=0).

Un seul PIPE:

Principe:

On crée un tableau “int f1[2];”
On fait le pipe “pipe(f1);”
On fait un premier fork:
Dans le fils:
On ferme la sortie standard “close(1);”
Et on le remplace par f1[1] “dup(f1[1]);”
On exécute la 1er commande!
Dans le père:
On fait le 2eme fork:
Dans le fils:
On ferme la sortie standard “close(0);”
Et on le remplace par f1[0] “dup(f1[0]);”
On exécute la 2eme commande!
Dans le père:
On attend la mort des fils “while (wait(NULL)!=-1);”
On Close f1 “close(f1[1]); close(f1[0]);”

Pour plus de détails on vous invite à voir le fichier “OnePipe.c” qui contient cette méthode.
Dans la version Finale de projet cette méthode n’est pas utilisée car “n_pipe” est généralisé!

Plusieurs PIPEs:

Principe:

| On crée un tableau de 2 dimensions : 2 cases pour chaque tube. “int tube[**nbc-1**][2];”

| On fait un parcours pour exécuter chaque commandes “for(i=0;i<**nbc**;i++)”

| **nbc** vaut le nombre de commande à exécuter donc **nbc-1** nombre de pipes qui |existes => on a besoin de nbc-1 tubes.

| Si on n’est pas à la dernière commande on fait un pipe “pipe(tube[i]);”

| Puis le fork

| Dans le fils:

| si on n'est pas à la 1er commande on recopie le descripteur |tube[i-1][0] dans le descripteur de l'entrée standard

| si on n'est pas à la dernière commande on recopie le descripteur |tube[i][0] dans le descripteur de la sortie standard

| On exécute la i eme commande!

Dans le père:

| On attend la mort des fils “while (wait(NULL)!=-1);”

| On ferme les tubes

Pour plus de détails on vous invite à voir le fichier “shell.c” qui contient cette méthode “n_pipe”.

6. Gestions des redirections:

Entrée:

Au niveau de fils:

Principe:

si une redirection < existe:
-> on met bin a 1.
-> et le nom de fichier dans sin.
puis on essaye d’ouvrir le fichier sin dans fd0
Si tout se passe bien on duplique fd0 dans 0
finalement on close fd0

Code:

```
int fd0;
if(bin){
  if ((fd0 = open(sin, O_RDONLY, 0)) < 0) {
       perror("Open <");exit(0);
  }
  dup2(fd0, 0);close(fd0);
}
```

Au niveau de père:

Principe:

stdin reprend sa place!

Code:

```
dup2(stdin,0);
```

Sortie:

Au niveau de fils:

Principe:

si une redirection > existe:

-> on met bout a 1.

-> et le nom de fichier dans sout.

puis on essaye d'ouvrir le fichier sout dans fd1

Si tout se passe bien on duplique fd1 dans 1

finalement on close fd1

Code:

```
int fd1;
if(bout){
  if ((fd1 = open(sout,O_CREAT|O_WRONLY, 0644)) < 0) {
     perror("Open >");exit(0);
  }
  dup2(fd1,1); close(fd1);
}
```

Au niveau de père:

Principe:

stdout reprend sa place!

Code:

```
dup2(stdout,0);
```

7. Tests:

Durant ce projet on effectuer plusieurs tests au fur a mesur de notre avancement pour tester toutes les commandes possibles:
Les premiers tests etait des commandes simple (ie : sans pipes, ni redirections ni “&” et ni “;”) et voici quelque tests:

- **test sur cd avec et sans paramètres:**

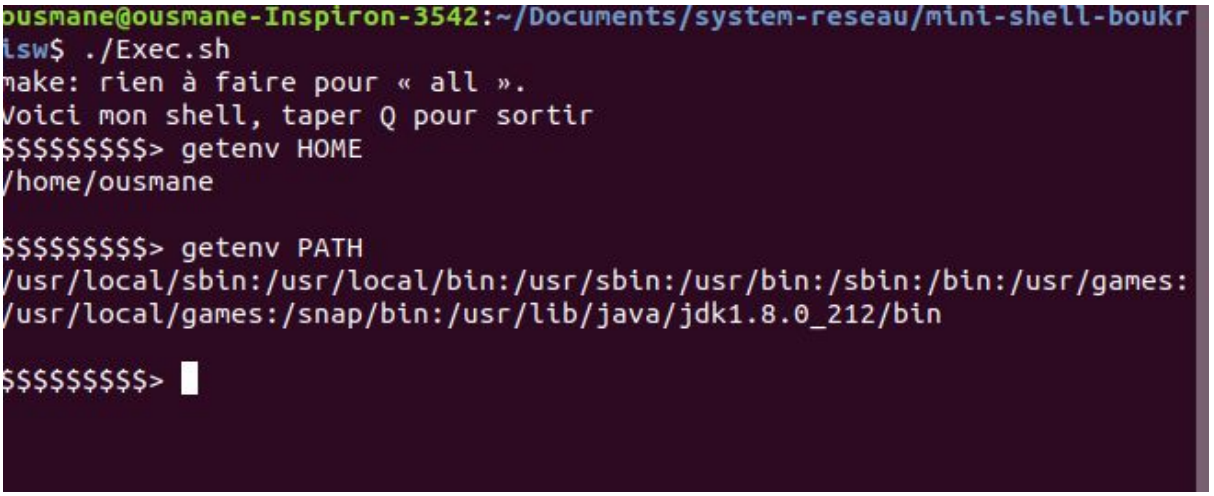
```
ousmane@ousmane-Inspiron-3542:~/Documents/system-reseau/mini-shell-boukr
isw$ ./Exec.sh
make: rien à faire pour « all ».
Voici mon shell, taper Q pour sortir
$$$$$$$$$> ls
cmd_simple    Exec.sh  Makefile   readcmd.h  shell    shell.o
cmd_simple.C  main.c   OnePipe.c  README.md  shell.c

$$$$$$$$$> cd

$$$$$$$$$> ls
Bureau            Images                      Musique          snap
Documents         jdk-8u212-linux-x64.tar.gz  NetBeansProjects  Téléchar
gements
examples.desktop  Modèles                     Public           Vidéos

$$$$$$$$$>
```

```
$$$$$$$$$> ls
Bureau            Images                      Musique          snap
Documents         jdk-8u212-linux-x64.tar.gz  NetBeansProjects  Téléchar
gements
examples.desktop  Modèles                     Public           Vidéos

$$$$$$$$$> cd Documents

$$$$$$$$$> ls
Algo            l3_miage.odt      LaTruiteOpaque  Sokoban
Document-1.jpg  l3_miage.pdf      logement.odt    system-reseau
fanorona        LangagePourLeWeb  Pousseur.jpeg

$$$$$$$$$>
```

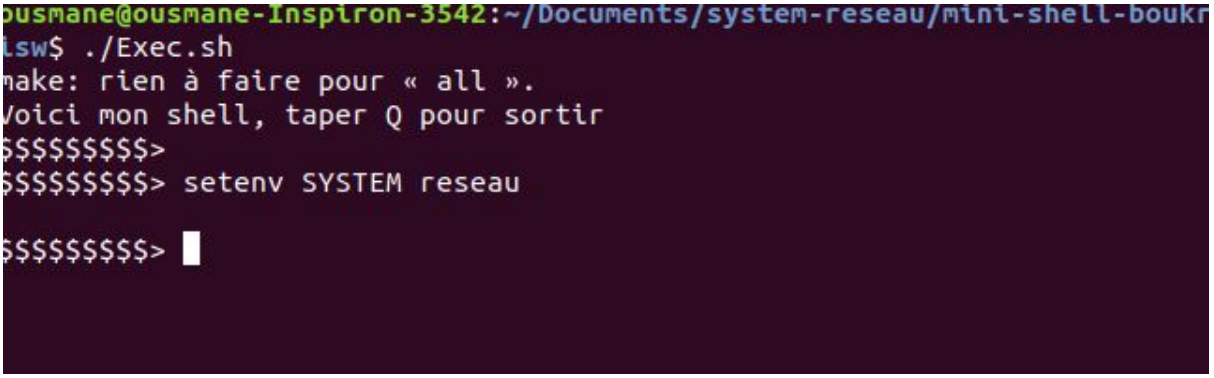
- **test sur exit**

```
$$$$$$$$$> cd Documents

$$$$$$$$$> ls
Algo            l3_miage.odt      LaTruiteOpaque  Sokoban
Document-1.jpg  l3_miage.pdf      logement.odt    system-reseau
fanorona        LangagePourLeWeb  Pousseur.jpeg

$$$$$$$$$> exit
ousmane@ousmane-Inspiron-3542:~/Documents/system-reseau/mini-shell-boukr
isw$
```

- **test sur getenv:**



- **test sur setenv avec et sans paramètre :**



```
$$$$$$$$> setenv
GJS_DEBUG_TOPICS=JS ERROR;JS LOG
LESSOPEN=| /usr/bin/lesspipe %s
USER=ousmane
TEXTDOMAIN=im-config
XDG_SEAT=seat0
SSH_AGENT_PID=1438
XDG_SESSION_TYPE=x11
SHLVL=1
QT4_IM_MODULE=xim
HOME=/home/ousmane
OLDPWD=/home/ousmane/Documents/system-reseau
DESKTOP_SESSION=ubuntu
GNOME_SHELL_SESSION_MODE=ubuntu
GTK_MODULES=gail:atk-bridge
DBUS_SESSION_BUS_ADDRESS=unix:path=/run/user/1000/bus
COLORTERM=truecolor
```

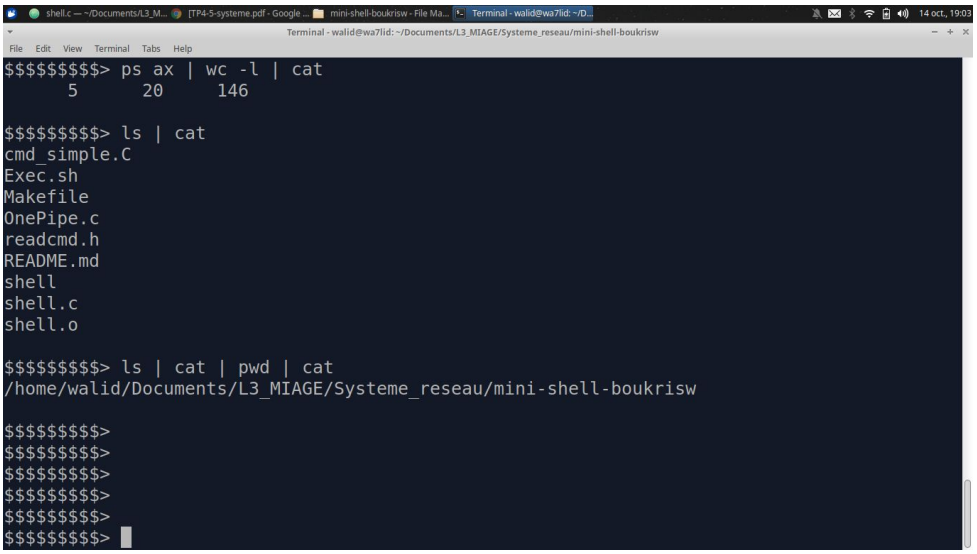
- **test sur plusieurs commandes séparés par des “;” :**

```
$$$$$$$$> pwd ; cd Documents ; ls
/home/ousmane

Algo            l3_miage.odt      LaTruiteOpaque  Sokoban
Document-1.jpg  l3_miage.pdf      logement.odt    system-reseau
fanorona        LangagePourLeWeb  Pousseur.jpeg

$$$$$$$$>
```

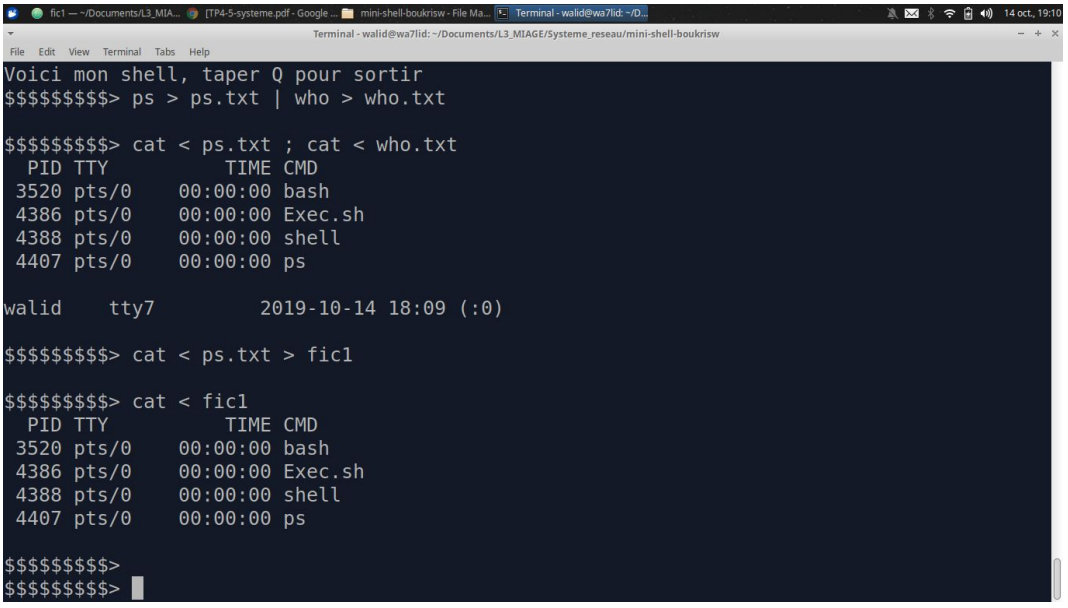
-

-

- **test sur deux commandes avec des pipes :**



- **test sur deux commandes avec des pipes et redirections :**



8. Conclusion:

Réaliser ce projet nous a apportée des avantages qui sont entre autre:

-Améliorer nos compétences en C, et bien maîtriser la gestion des tableaux et l'allocation en mémoire.

-Bien s'organiser en binôme et se partager les tâches, et respecter le cahier de charges et respecter les délais.

-bonne utilisation de git, makefile et script shell.

-Bonnes connaissances des commandes shell et leurs fonctions!