



# **LO14 – ADMINISTRATION DES SYSTÈMES**

Rapport final du projet

EI MCHANTEF Safouane / KOUROUMA Idriss

Automne

## Table des matières

Introduction :.....	2
Structure :.....	2
1) Mode List .....	3
2) Mode create .....	3
3) Mode extract .....	4
4) Mode Browse .....	5
pwd & cd .....	5
Ls .....	5
Cat .....	6
touch & mkdir .....	7
Rm .....	8

# Introduction :

Dans le cadre de l'UE LO14 - Administration des systèmes, nous étions amenés à réaliser un projet ayant pour objectif de créer un serveur d'archive.

Ce projet est réalisé en **bash sur Linux**.

Le but est de pouvoir communiquer avec ce serveur à l'aide d'une nouvelle commande shell, nommée vsh. Cette commande, accompagnée d'options (vsh -option [nom\_serveur] [port]) permet au client :

- De dresser la liste des archives
- De créer une archive
- D'explorer le serveur d'archive
- D'extraire le contenu d'une archive dans le répertoire courant de la machine

Dans ce rapport, nous présentons la structure finale du projet, et l'analyse des différentes fonctions par rapport à nos premières idées.

Notre code est disponible sur : <https://github.com/ElSafouane/projetLO14>

## Structure :

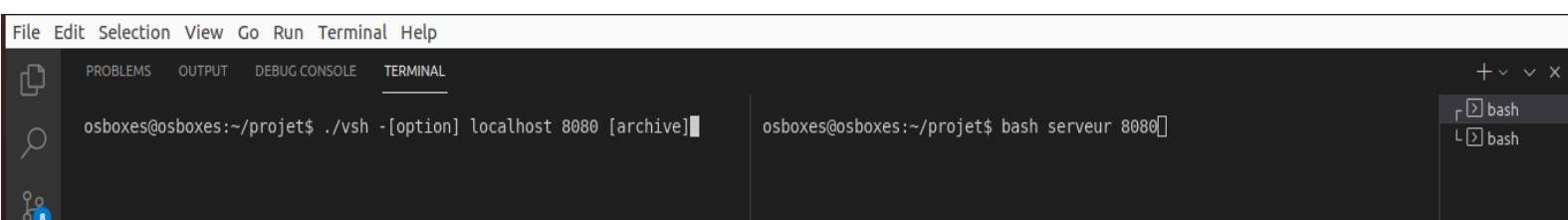
On a utilisé comme structure de base de notre serveur l'exemple de **serveur netcat** vu en TD.

NetCat permet d'établir une connexion client / serveur. Ce que le client lit sur son entrée standard est envoyé au serveur et ce que le serveur envoie est lu sur la sortie standard du client. Pour se connecter au serveur il faut préciser les options -l (le serveur est en mode écoute) et -p pour préciser le port.

La fonction interaction présente dans le script serveur écrit les réponses aux requêtes du client sur sa sortie standard. C'est celle qui permet au client d'écrire les commandes qu'il veut exécuter.

L'arborescence de notre archive utilise comme répertoire racine le "\\" ou backslash, les fichiers seront alors tous localisés à partir de ce répertoire racine et auront un chemin absolu ressemblant à l'exemple : "\\Exemple\Test\A"

Une fois l'archive créée avec les \, le client doit utiliser \ pour explorer celle-ci.



```
File Edit Selection View Go Run Terminal Help
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
osboxes@osboxes:~/projet$ ./vsh -[option] localhost 8080 [archive]
osboxes@osboxes:~/projet$ bash serveur 8080
bash
bash
```

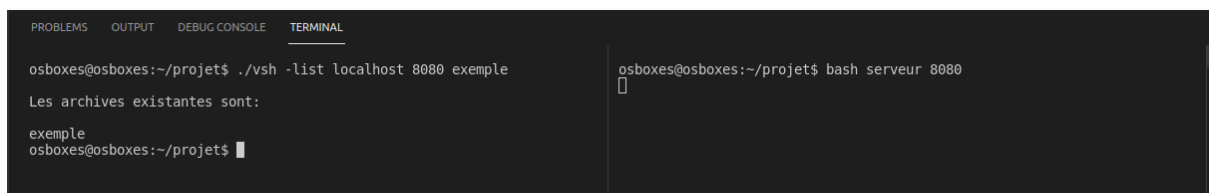
## 1) Mode List

Dans un premier temps, le mode list de notre serveur d'archive permet d'afficher sur la sortie standard du client, la liste des archives présentes sur le serveur.

Toutes les archives sont stockées dans un répertoire, que le client appellera pour avoir connaissance de toutes celles présentes sur le serveur.

Cette fonction est implémentée à partir de la commande `vsh -list nom_serveur port` avec `nom_serveur` l'adresse du serveur et `port` le numéro de port sur lequel le serveur attend une requête.

Lorsque le client appelle cette commande, le nom des archives préalablement stockées dans un répertoire archive est affiché à l'écran.



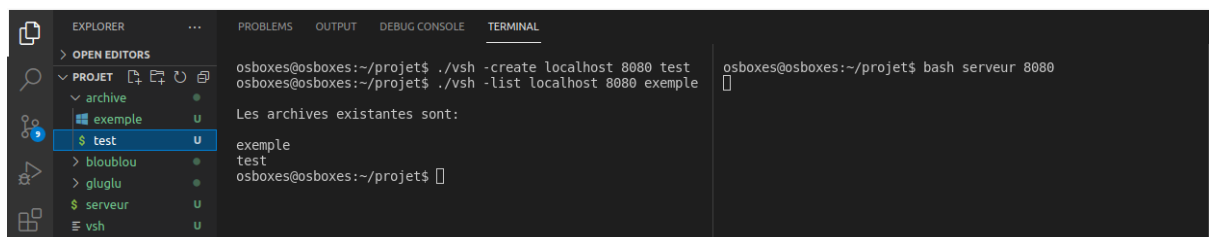
```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
osboxes@osboxes:~/projet$ ./vsh -list localhost 8080 exemple
Les archives existantes sont:
exemple
osboxes@osboxes:~/projet$
```

```
osboxes@osboxes:~/projet$ bash serveur 8080
```

## 2) Mode create

Le mode create nous permet de créer une archive dans le serveur à partir du répertoire courant du client. L'option choisie pour le mode create a été d'intégrer celle-ci au niveau du script `vsh` réservée au client et non sur le script `serveur` pour faciliter la transmission d'informations. Ainsi l'archive est dans un premier temps créée au niveau du client sur sa machine personnelle avant de transmettre ce fichier texte au serveur.

Le serveur aura pour unique fonction de recevoir ce fichier et de le déplacer dans le répertoire archive dédié au stockage des archives.



```
EXPLORER
> OPEN EDITORS
PROJET
  archive
    exemple
    $ test
    > bloublou
    > gluglu
    $ serveur
    vsh

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
osboxes@osboxes:~/projet$ ./vsh -create localhost 8080 test
osboxes@osboxes:~/projet$ ./vsh -list localhost 8080 exemple
Les archives existantes sont:
exemple
test
osboxes@osboxes:~/projet$
```

```
osboxes@osboxes:~/projet$ bash serveur 8080
```

*Exécution du mode create. Archive test créée dans le dossier archive.*

```

1  #! /bin/bash
2
3  directory \home\osboxes\projet\
4  archive drwxrwxr-x 4096
5  bloublou drwxrwxr-x 4096
6  gluglu drwxrwxr-x 4096
7  serveur -rw-rw-r-- 20309 1 817
8  vsh -rwxrwxr-x 7814 818 309
9
10 directory \home\osboxes\projet\archive
11 exemple -rw-rw-r-- 871 1127 45
12 @
13 directory \home\osboxes\projet\bloublou
14 gjfkd -rw-rw-r-- 8 1172 1
15 @
16 directory \home\osboxes\projet\gluglu
17 agua -rw-rw-r-- 0 1173
18 gluglu2 drwxrwxr-x 4096
19 @
20 directory \home\osboxes\projet\gluglu\gluglu2
21 agua2 -rw-rw-r-- 12 1173 2
22 gluglu3 drwxrwxr-x 4096
23 @
24 directory \home\osboxes\projet\gluglu\gluglu2\gluglu3
25 eau3 -rw-rw-r-- 4 1175
26 @
27 #! /bin/bash
28
29 # Ce script implémente un serveur.
30 # Le script doit être invoqué avec l'argument :
31 # PORT le port sur lequel le serveur attend ses clients
32
33 if [ $# -ne 1 ]; then
34     echo "usage: $(basename $0) PORT"
35     exit -1
36 fi
37
38 PORT=$1

```

Capture d'écran de l'archive test. Le header débute ligne 3 et le body débute ligne 27.

### 3) Mode extract

Le mode extract doit réaliser la fonction inverse de create, c'est-à-dire de faire passer une archive sous le format d'un fichier texte du serveur à une arborescence réelle au niveau du client.

Ainsi le client devra exécuter la fonction extract sur une archive préalablement existante dans le serveur pour la recevoir.

Les permissions sont respectées lors du transfert, l'utilisateur ne pourra pas exécuter un script s'il était en lecture seule dans le fichier texte de l'archive.

```

osboxes@osboxes:~/projet$ ./vsh -extract localhost 8080 exemple
osboxes@osboxes:~/projet$ bash serveur 8080

```

Mode extract : l'archive exemple est extraite au niveau du répertoire courant

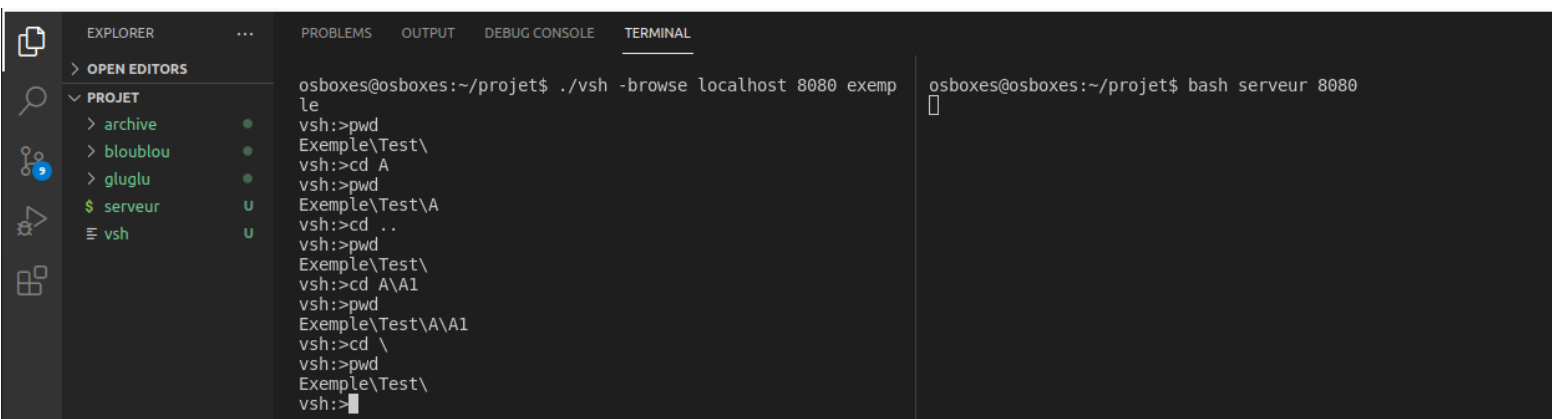
## 4) Mode Browse

Le mode Browse est constitué de plusieurs commandes afin de naviguer dans notre archive comme dans une archive classique. Lorsque l'on lance le mode browse la connexion ne se ferme pas entre le client et le serveur et on peut exécuter certaines fonctions.

### pwd & cd :

La commande pwd nous indique le répertoire courant dans lequel se trouve l'utilisateur. La commande cd permet elle de se déplacer d'un répertoire à un autre.

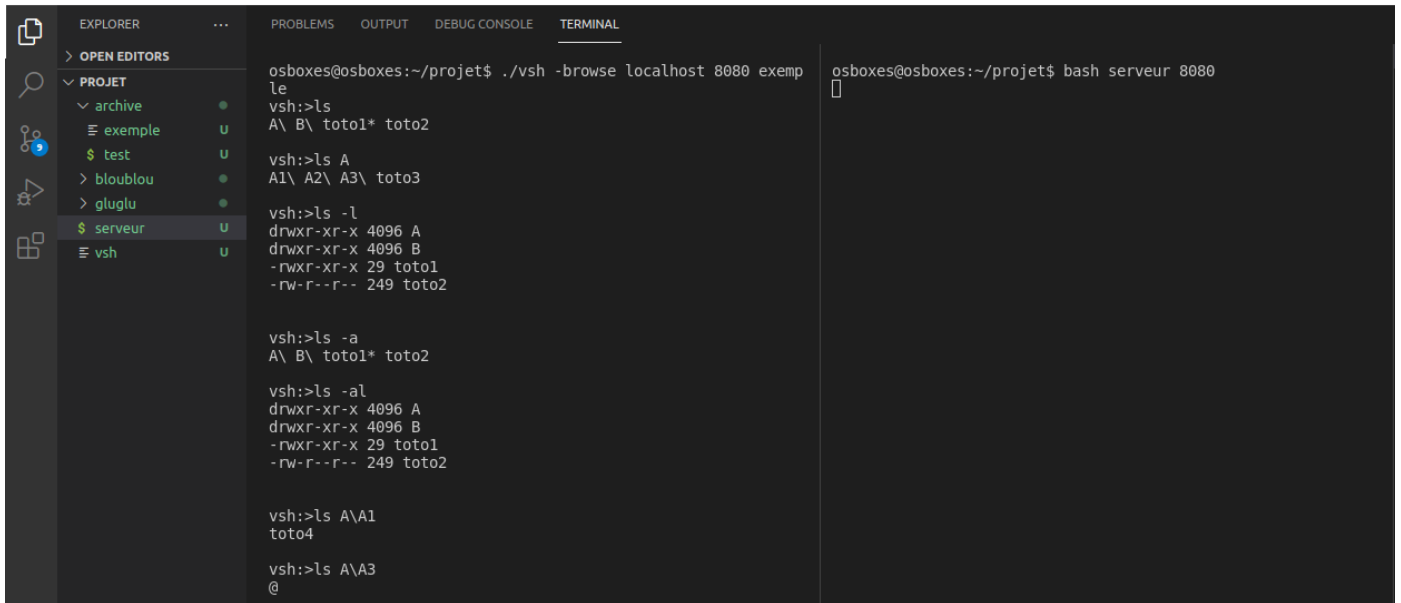
Le client pourra exécuter la commande "cd \" afin de revenir dans le répertoire racine de l'archive. Il pourra également se déplacer en arrière à l'aide de la commande "cd .." ou encore se déplacer dans plusieurs dossiers en même temps en indiquant "cd A\B\C".



The screenshot shows a VS Code editor with a terminal window open. The terminal displays a vsh shell session. The user has entered the command `./vsh -browse localhost 8080 exemple` and is now in a vsh prompt. The user has entered several commands: `pwd` (returns `Exemple\Test\`), `cd A` (returns `Exemple\Test\A`), `cd ..` (returns `Exemple\Test\`), `cd A\A1` (returns `Exemple\Test\A\A1`), and `cd \` (returns `Exemple\Test\`). The user is currently at the vsh prompt.

### Ls :

La commande ls permet comme sur un shell classique de lister les fichiers présents dans un dossier. On peut également faire la liste des fichiers d'un dossier sans obligatoirement se trouver dedans avec la commande "ls A\A1" par exemple. La commande "ls -l" permet d'afficher les fichiers sous forme de liste et le "ls -a" permet d'afficher les fichiers cachés également, il est possible d'exécuter la commande "ls -al".



```
osboxes@osboxes:~/projet$ ./vsh -browse localhost 8080 exemp
le
vsh:>ls
A\ B\ toto1* toto2

vsh:>ls A
A1\ A2\ A3\ toto3

vsh:>ls -l
drwxr-xr-x 4096 A
drwxr-xr-x 4096 B
-rwxr-xr-x 29 toto1
-rw-r--r-- 249 toto2

vsh:>ls -a
A\ B\ toto1* toto2

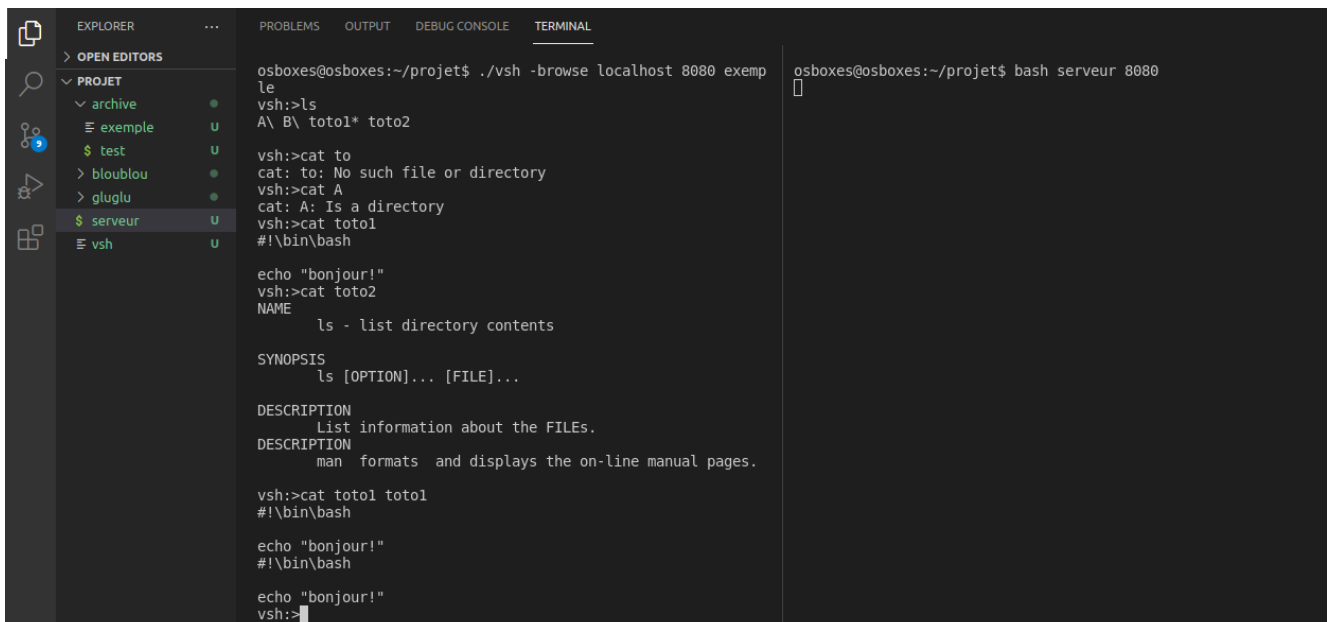
vsh:>ls -al
drwxr-xr-x 4096 A
drwxr-xr-x 4096 B
-rwxr-xr-x 29 toto1
-rw-r--r-- 249 toto2

vsh:>ls A\A1
toto4

vsh:>ls A\A3
@
```

## Cat :

La commande cat permet d'accéder au contenu d'un fichier de l'archive. On peut également



```
osboxes@osboxes:~/projet$ ./vsh -browse localhost 8080 exemp
le
vsh:>ls
A\ B\ toto1* toto2

vsh:>cat to
cat: to: No such file or directory
vsh:>cat A
cat: A: Is a directory
vsh:>cat toto1
#!\bin\bash

echo "bonjour!"
vsh:>cat toto2
NAME
    ls - list directory contents

SYNOPSIS
    ls [OPTION]... [FILE]...

DESCRIPTION
    List information about the FILES.
DESCRIPTION
    man formats and displays the on-line manual pages.

vsh:>cat toto1 toto1
#!\bin\bash

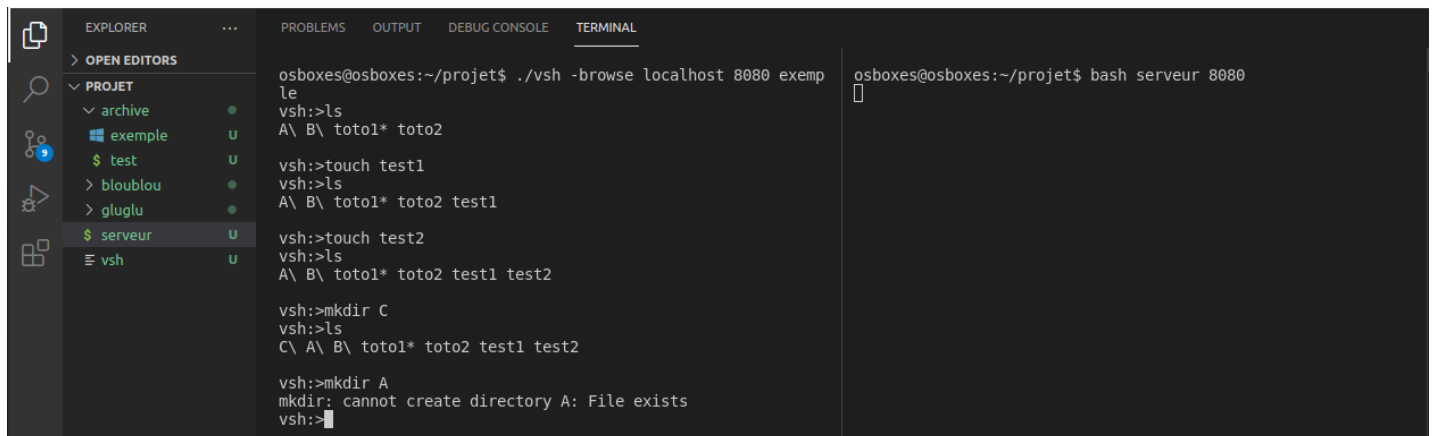
echo "bonjour!"
#!\bin\bash

echo "bonjour!"
vsh:>
```

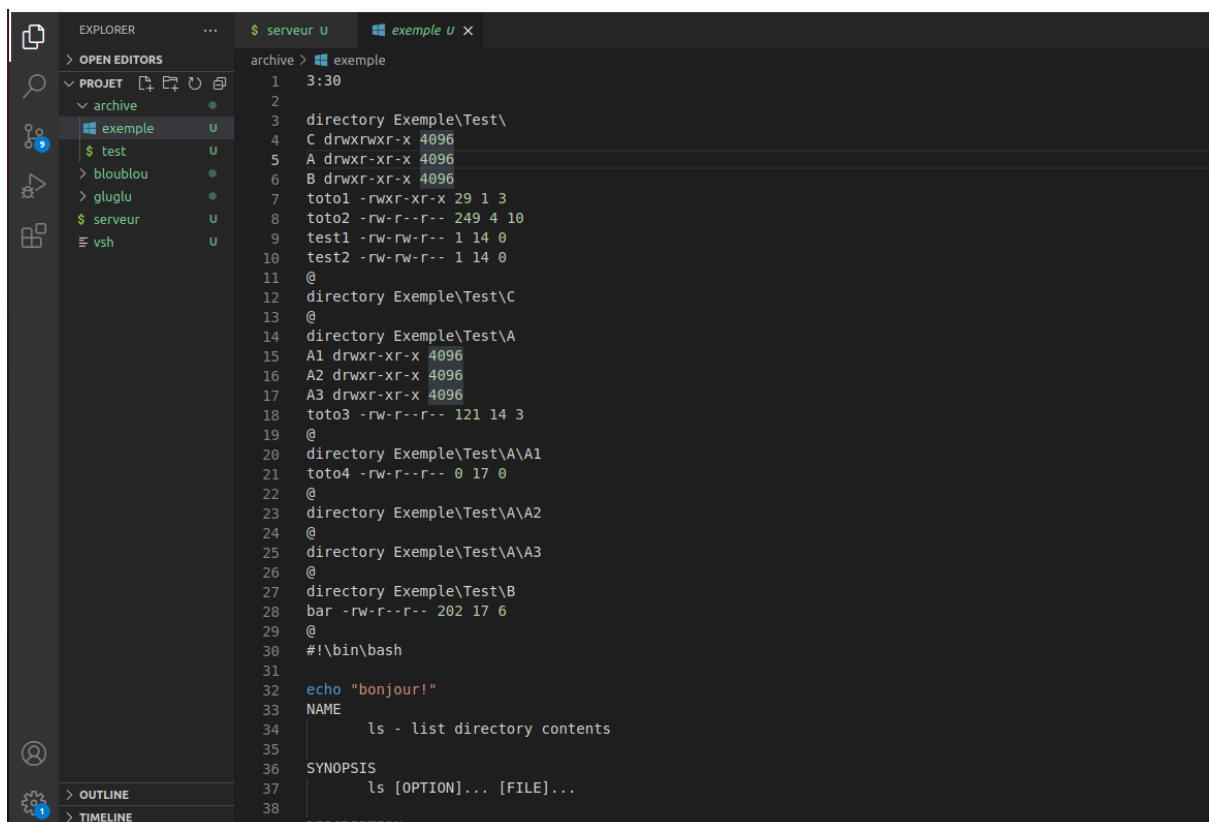
## Touch & mkdir :

La commande touch permet de créer un fichier vide dans l'archive au niveau du répertoire où l'on se trouve.

La commande mkdir permet de créer un dossier vide dans l'archive. Ce dossier se crée sous la forme "directory Exemple\Test\NouveauDossier" et également sous forme de ligne "NouveauDossier drwxrwxr-x 4096" au niveau supérieur.



```
osboxes@osboxes:~/projet$ ./vsh -browse localhost 8080 exemple
vsh:>ls
A\ B\ toto1* toto2
vsh:>touch test1
vsh:>ls
A\ B\ toto1* toto2 test1
vsh:>touch test2
vsh:>ls
A\ B\ toto1* toto2 test1 test2
vsh:>mkdir C
vsh:>ls
C\ A\ B\ toto1* toto2 test1 test2
vsh:>mkdir A
mkdir: cannot create directory A: File exists
vsh:>
```



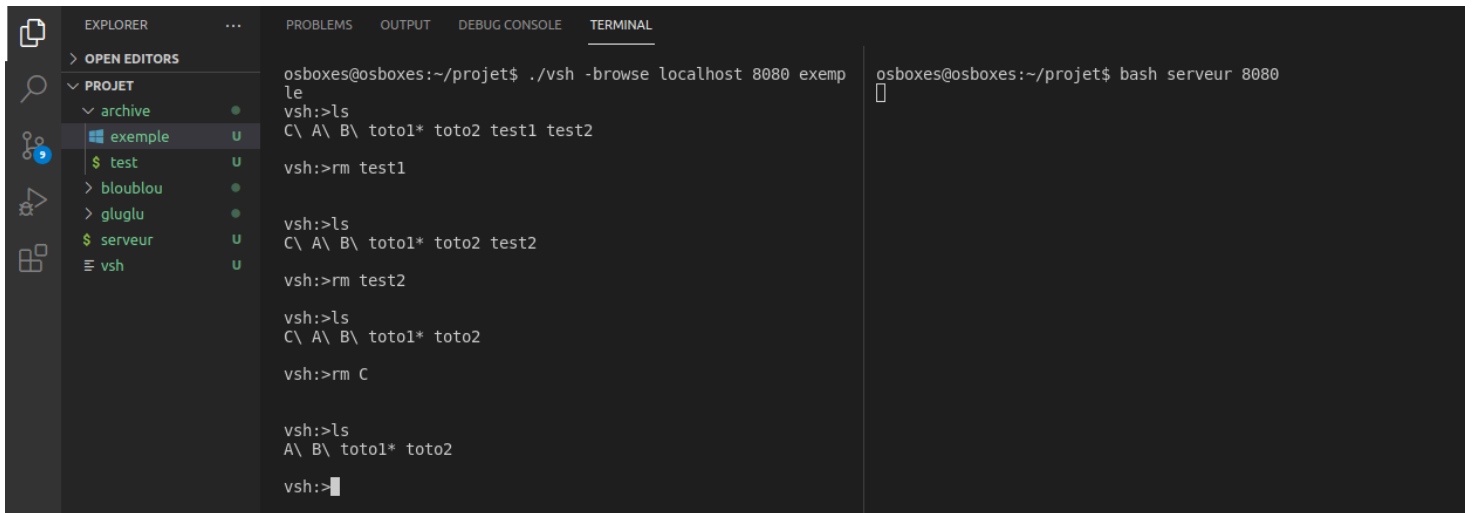
```
archive > exemple
1 3:30
2
3 directory Exemple\Test\
4 C drwxrwxr-x 4096
5 A drwxr-xr-x 4096
6 B drwxr-xr-x 4096
7 toto1 -rwxr-xr-x 29 1 3
8 toto2 -rw-r--r-- 249 4 10
9 test1 -rw-rw-r-- 1 14 0
10 test2 -rw-rw-r-- 1 14 0
11 @
12 directory Exemple\Test\C
13 @
14 directory Exemple\Test\A
15 A1 drwxr-xr-x 4096
16 A2 drwxr-xr-x 4096
17 A3 drwxr-xr-x 4096
18 toto3 -rw-r--r-- 121 14 3
19 @
20 directory Exemple\Test\A\A1
21 toto4 -rw-r--r-- 0 17 0
22 @
23 directory Exemple\Test\A\A2
24 @
25 directory Exemple\Test\A\A3
26 @
27 directory Exemple\Test\B
28 bar -rw-r--r-- 202 17 6
29 @
30 #!\bin\bash
31
32 echo "bonjour!"
33 NAME
34 ls - list directory contents
35
36 SYNOPSIS
37 ls [OPTION]... [FILE]...
38
39 DESCRIPTION
```

*Archive mis-à-jour après les différentes création de fichiers et de répertoires*



## Rm :

Le rm permet de supprimer des fichiers ou des dossiers dans l'archive.



The screenshot shows the Visual Studio Code interface with the Explorer, Search, and Run and Debug views on the left. The Explorer view shows a project structure with folders 'archive', 'exemple', 'bloublou', 'gluglu', 'serveur', and 'vsh'. The 'exemple' folder is selected. The Search view shows results for 'test' and 'serveur'. The Run and Debug view shows a terminal session with vsh commands. The terminal output is as follows:

```
osboxes@osboxes:~/projet$ ./vsh -browse localhost 8080 exemple
ls
vsh:>ls
C\ A\ B\ toto1* toto2 test1 test2
vsh:>rm test1
vsh:>ls
C\ A\ B\ toto1* toto2 test2
vsh:>rm test2
vsh:>ls
C\ A\ B\ toto1* toto2
vsh:>rm C
vsh:>ls
A\ B\ toto1* toto2
vsh:>
```

The terminal session shows the user running a series of commands to browse and manage files in a virtual file system. The commands and their outputs are: `./vsh -browse localhost 8080 exemple` (output: `ls`), `vsh:>ls` (output: `C\ A\ B\ toto1* toto2 test1 test2`), `vsh:>rm test1`, `vsh:>ls` (output: `C\ A\ B\ toto1* toto2 test2`), `vsh:>rm test2`, `vsh:>ls` (output: `C\ A\ B\ toto1* toto2`), `vsh:>rm C`, `vsh:>ls` (output: `A\ B\ toto1* toto2`), and `vsh:>`.