

LO03 Rapport de Projet

Hiver 2022-2023

SYNCHRONISEUR DE SYSTEMES DE FICHIERS SOUS LINUX

Université de technologie sino-européenne de Shanghai

(UTSEUS)



ChenJie LU 20124561

YiLin YANG 20124704

26 Février 2023

Catalogue

1. Aperçu du projet.....	1
1.1 Présentation du Projet	1
1.2 La forme de synchronisation.....	1
2. Processus du projet.....	2
2.1 Créer le dossier de synchronisation <i>Repertoire_Partage</i> dans <i>UserA</i> et <i>UserB</i>	2
2.2 Exécuter le script dans le terminal	2
2.3 Entrer deux chemins de dossiers pour être synchronisés.....	2
2.4 Le synchroniseur commence à fonctionner	3
a. Copier.....	4
b. Supprimer.....	4
c. Modifier(gestion des conflits).....	4
d. Rectification d'une erreur.....	5
3. Implémentation du code.....	5
3.1 function <i>AddtoSynchro</i>	5
3.2 function <i>fiche_syn</i>	6-7
3.3 function <i>rep_syn</i>	8
3.4 function <i>SYNCHRO</i>	9
4. Division du travail.....	10
5. Améliorations et les autres.....	10
6. Réflexion	12
7. Référence	12
8. Annexe (code de projet complet)	13

Conseils:

Nous distinguons les fichiers, dossiers, fonctions, programmes de différentes couleurs dans le rapport pour une lecture facile.

Numéro	Couleur	Type
1	orange	Fichier
2	bleu	Dossier
3	vert	Fonction
4	violet	Programme

1. Aperçu du projet

1.1 Présentation du Projet

Dans ce projet, nous réalisons une synchronisation automatique du contenu (données et métadonnées) de deux dossiers utilisateur (*UserA* et *UserB*) en écrivant un script shell sur le système d'exploitation Linux et en l'exécutant dans le terminal. Les métadonnées font référence au type, aux permissions, à la taille et à la date de dernière modification d'un fichier ou dossier, nous enregistrons le journal de synchronisation dans *.synchro*, et nous donnons également des contre-mesures de conflit correspondantes, et effectuons certaines extensions à la fonction, telles qu'une interface conviviale et la synchronisation des liens symboliques.

1.2 La forme de synchronisation

a. Ajouter

Si un nouveau fichier/dossier est ajouté au dossier *UserA*, (le nom et les métadonnées du fichier/dossier ne sont pas sur *.synchro*) *«Synchroniseur»* ajoutera le copie dans le dossier *UserB* (y compris les permissions).

b. Supprimer

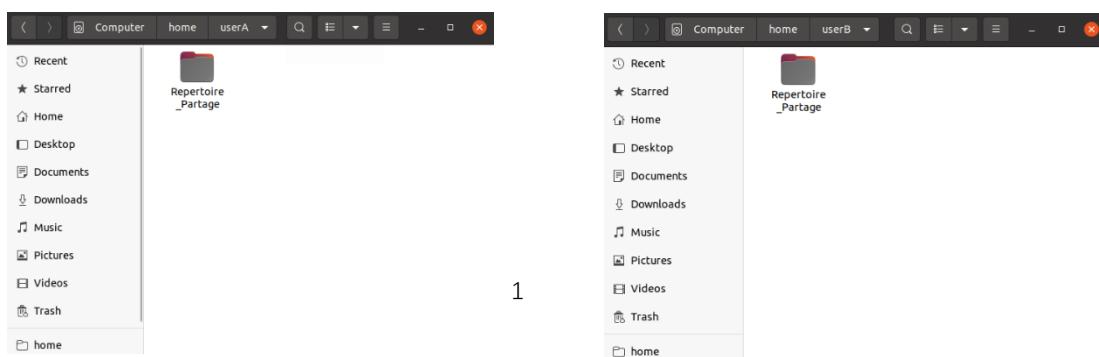
Si on supprime un fichier/dossier dans le dossier *UserA* (il a déjà existé sur *.synchro*) *«Synchroniseur»* supprimera également sa sauvegarde dans *UserB*.

c. Modifier

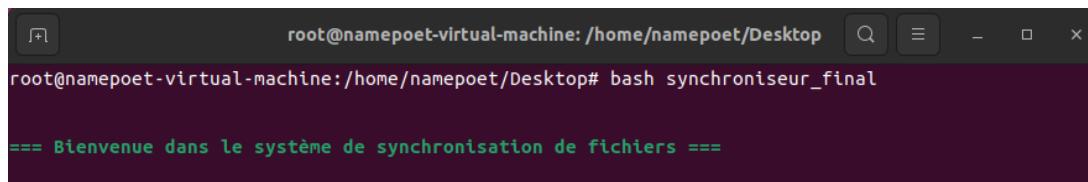
Si on change le contenu ou les permissions d'un fichier ou on renomme un dossier dans le dossier *UserA*, *«Synchroniseur»* fera la même chose dans le dossier *UserB*.

2. Processus du projet

2.1 Créer le dossier de synchronisation *Repertoire_Partage* dans *UserA* et *UserB*

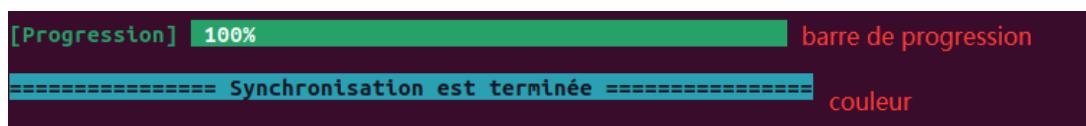


2.2 Exécuter le script dans le terminal

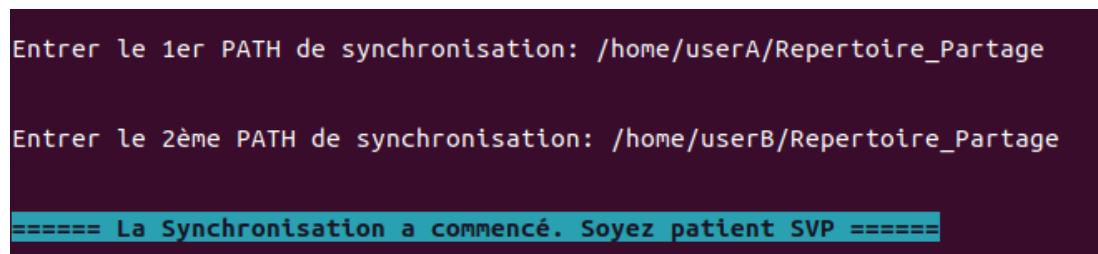


```
root@namepoet-virtual-machine:/home/namepoet/Desktop# bash synchroniseur_final
== Bienvenue dans le système de synchronisation de fichiers ==
```

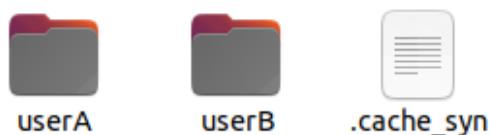
Nous avons conçu une interface conviviale qui permettent aux utilisateurs de comprendre facilement le processus d'exécution du programme (ex : différentes couleurs de suggestions, la barre de progression ...)



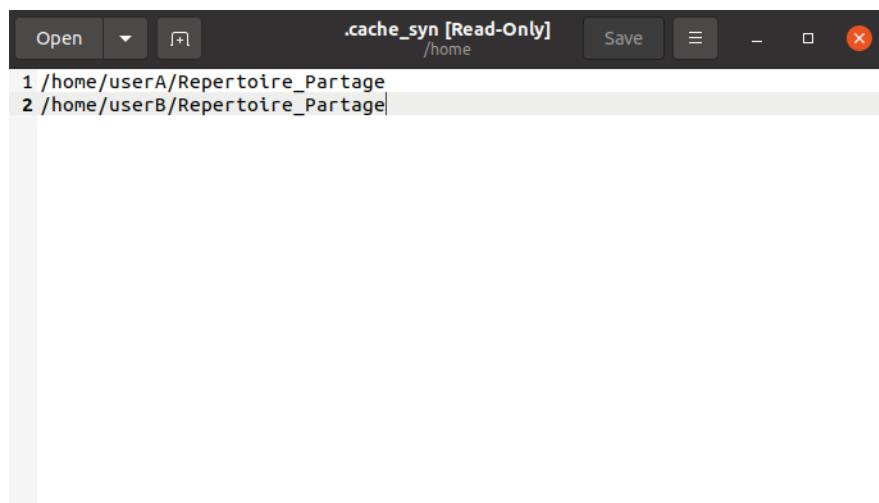
2.3 Entrer deux chemins de dossiers pour être synchronisés



```
Entrer le 1er PATH de synchronisation: /home/userA/Repertoire_Partage
Entrer le 2ème PATH de synchronisation: /home/userB/Repertoire_Partage
===== La Synchronisation a commencé. Soyez patient SVP =====
```



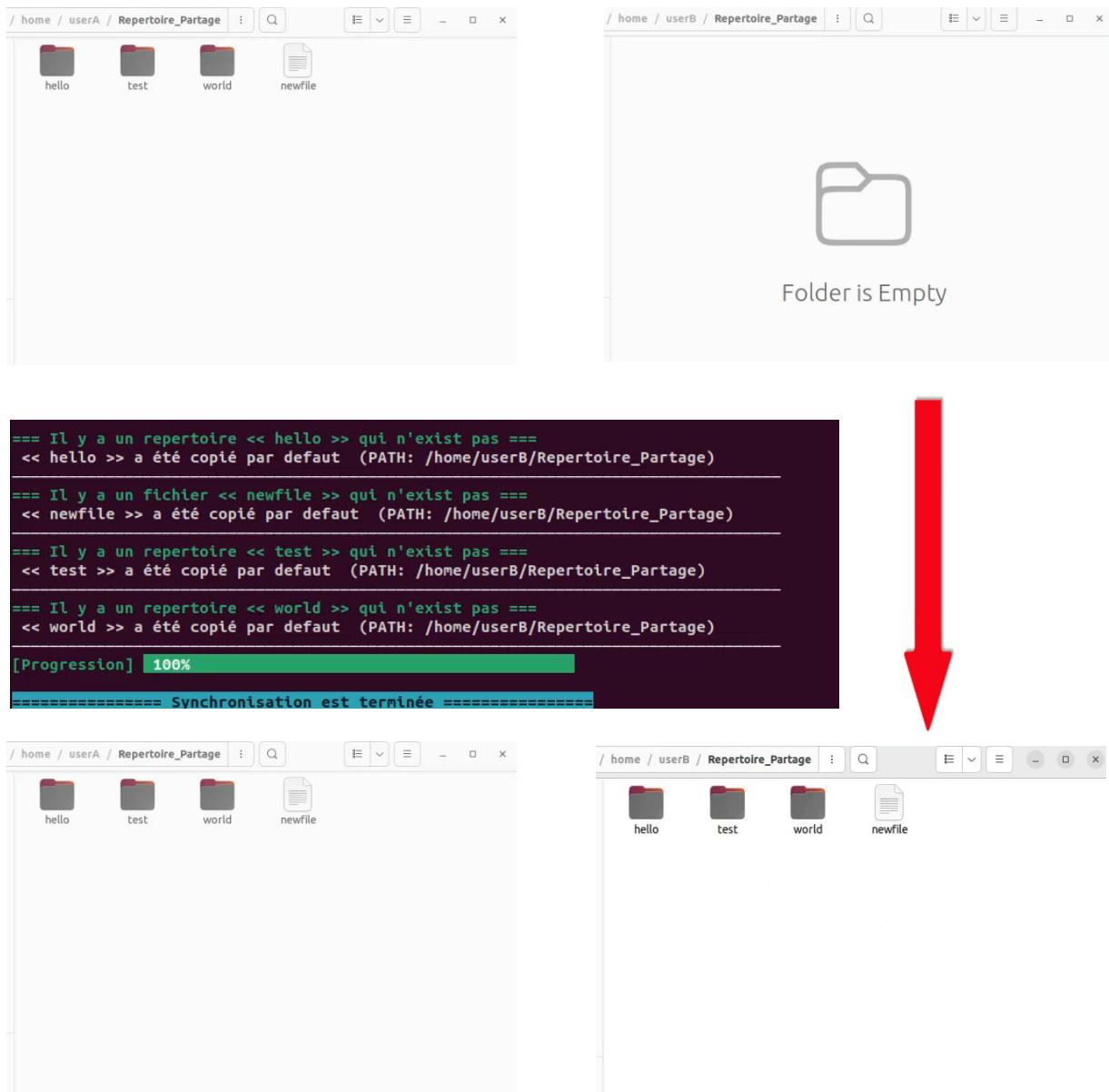
Les deux chemins sont stocké dans le fichier *.cache_syn*



Open ▾ + .cache_syn [Read-Only] /home Save ⌂ ×

```
1 /home/userA/Repertoire_Partage
2 /home/userB/Repertoire_Partage|
```

2.4 Le synchroniseur commence à fonctionner



Le terminal affiche un message pour la synchronisation du fichier ou du dossier, ainsi qu'un marque pour le copier, le supprimer ou le modifier, et le chemin à la synchronisation.

a. Copier

```

    === Il y a un fichier << add >> qui n'existe pas ===
    << add >> a été copié par défaut (PATH: /home/userA/Repertoire_Partage)
  
```

b. Supprimer

```

    === Le fichier << new >> a été supprimé par l'utilisateur ===
    << new >> a été supprimé par défaut (PATH:/home/userA/Repertoire_Partage)
  
```

c. Modifier (Gestion des conflits)

```
=====
Il y a un conflit =====
Raison: Les DEUX parties ont modifié ou créé <> addAdd > avant la synchronisation

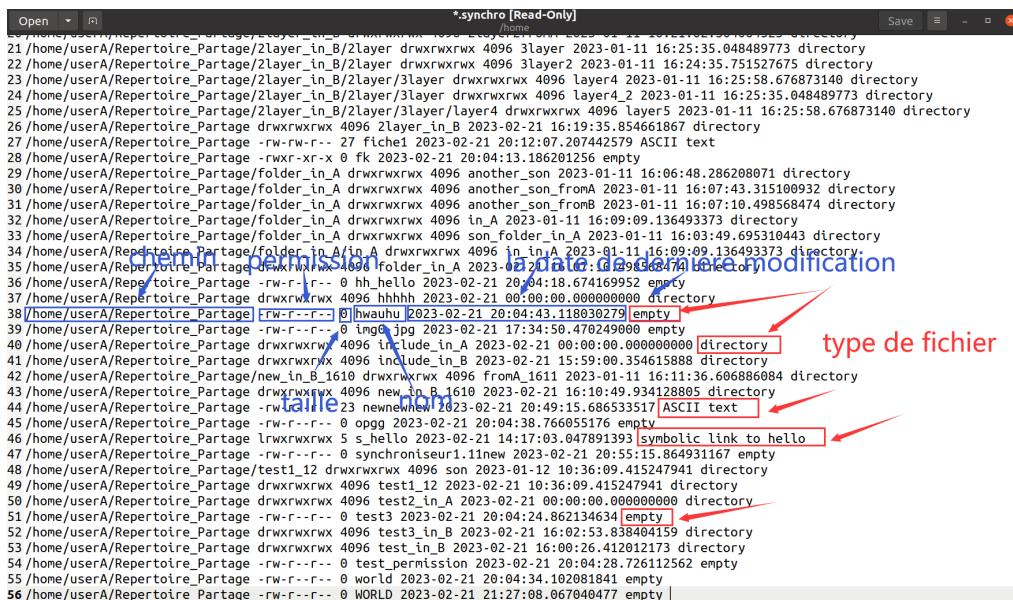
[1] PATH:/home/userA/Repertoire_Partage:
1 ajouter
2
3
4 modifier
5
6
7 assas

[2] PATH:/home/userB/Repertoire_Partage:
1 ajouter
2
3
4 modifier
5
6
7 add

=====
Le fichier de quelle partie souhaitez-vous conserver? [1]/[2]2

== << addAdd >> de PATH:/home/userB/Repertoire_Partage a été conservé ==
[Progression] 100%
===== Synchronisation est terminée =====
```

Le terminal affiche le contenu du fichier en conflit et demande à l'utilisateur quelle version à synchroniser. (*L'utilisateur peut entrer 1 ou 2*)



```
*synchro [Read-Only]
/home
21/home/userA/Repertoire_Partage/layer_in_B/2layer drwxrwxrwx 4096 3layer 2023-01-11 16:25:35.048489773 directory
22/home/userA/Repertoire_Partage/layer_in_B/2layer drwxrwxrwx 4096 3layer2 2023-01-11 16:24:35.751527675 directory
23/home/userA/Repertoire_Partage/layer_in_B/2layer/3layer drwxrwxrwx 4096 3layer4 2023-01-11 16:25:58.676873140 directory
24/home/userA/Repertoire_Partage/layer_in_B/2layer/3layer drwxrwxrwx 4096 3layer4_2 2023-01-11 16:25:35.048489773 directory
25/home/userA/Repertoire_Partage/layer_in_B/2layer/3layer/layer4 drwxrwxrwx 4096 3layer5 2023-01-11 16:25:58.676873140 directory
26/home/userA/Repertoire_Partage drwxrwxrwx 4096 2layer_in_B 2023-02-21 16:19:35.854661867 directory
27/home/userA/Repertoire_Partage -rw-rw-r-- 27 fichier 2023-02-21 20:12:07.207442579 ASCII text
28/home/userA/Repertoire_Partage -rwxr-xr-x 0 fk 2023-02-21 20:04:13.186201256 empty
29/home/userA/Repertoire_Partage/folder_in_A drwxrwxrwx 4096 another_son 2023-01-11 16:06:48.286208071 directory
30/home/userA/Repertoire_Partage/folder_in_A drwxrwxrwx 4096 another_son_fromA 2023-01-11 16:07:43.315100932 directory
31/home/userA/Repertoire_Partage/folder_in_A drwxrwxrwx 4096 another_son_fromB 2023-01-11 16:07:10.498568474 directory
32/home/userA/Repertoire_Partage/folder_in_A drwxrwxrwx 4096 in_A 2023-01-11 16:09:09.136493373 directory
33/home/userA/Repertoire_Partage/folder_in_A drwxrwxrwx 4096 son_folder_in_A 2023-01-11 16:03:49.695310443 directory
34/home/userA/Repertoire_Partage/folder_in_A/in_A drwxrwxrwx 4096 in_in_A 2023-01-11 16:09:09.136493373 directory
35/home/userA/Repertoire_Partage drwxrwxrwx 4096 new_in_B 2023-02-21 20:44:18.674169952 empty
36/home/userA/Repertoire_Partage -rw-r--r-- 0 hh_hello 2023-02-21 20:44:18.674169952 empty
37/home/userA/Repertoire_Partage drwxrwxrwx 4096 hhhh 2023-02-21 00:00:00.000000000 directory
38/home/userA/Repertoire_Partage -rw-r--r-- 0 ihuahu 2023-02-21 20:04:43.118030279 empty
39/home/userA/Repertoire_Partage -rw-r--r-- 0 img0.jpg 2023-02-21 17:34:50.470249900 empty
40/home/userA/Repertoire_Partage drwxrwxrwx 4096 include_in_A 2023-02-21 00:00:00.000000000 directory
41/home/userA/Repertoire_Partage drwxrwxrwx 4096 include_in_B 2023-02-21 15:59:00.354615888 directory
42/home/userA/Repertoire_Partage/new_in_B_1610 drwxrwxrwx 4096 fromA_1611 2023-01-11 16:11:36.606886084 directory
43/home/userA/Repertoire_Partage drwxrwxrwx 4096 new_in_B_1610 2023-02-21 16:10:49.934128805 directory
44/home/userA/Repertoire_Partage -rw-r--r-- 23 nouveau 2023-02-21 20:49:15.686533517 ASCII text
45/home/userA/Repertoire_Partage -rw-r--r-- 0 oppg 2023-02-21 20:04:38.766055176 empty
46/home/userA/Repertoire_Partage lrwxrwxrwx 5 s_hello 2023-02-21 14:17:03.047891393 symbolic link to hello
47/home/userA/Repertoire_Partage -rw-r--r-- 0 synchroniseur1.linen 2023-02-21 20:55:15.864931167 empty
48/home/userA/Repertoire_Partage/test1_12 drwxrwxrwx 4096 son 2023-01-12 10:36:09.415247941 directory
49/home/userA/Repertoire_Partage drwxrwxrwx 4096 test1_12 2023-02-21 10:36:09.415247941 directory
50/home/userA/Repertoire_Partage drwxrwxrwx 4096 test2_in_A 2023-02-21 00:00:00.000000000 directory
51/home/userA/Repertoire_Partage -rw-r--r-- 0 test3 2023-02-21 20:04:24.862134634 empty
52/home/userA/Repertoire_Partage drwxrwxrwx 4096 test3_in_B 2023-02-21 16:02:53.838404159 directory
53/home/userA/Repertoire_Partage drwxrwxrwx 4096 test_in_B 2023-02-21 16:00:26.412012173 directory
54/home/userA/Repertoire_Partage -rw-r--r-- 0 test_permission 2023-02-21 20:04:28.726112562 empty
55/home/userA/Repertoire_Partage -rw-r--r-- 0 world 2023-02-21 20:04:34.102081841 empty
56/home/userA/Repertoire_Partage -rw-r--r-- 0 WORLD 2023-02-21 21:27:08.067040477 empty |
```

Le journal de synchronisation est écrit dans le fichier *.synchro*. Le chemin, les permissions, la taille, la date de modification et le type de fichier (*directory*, *empty*, *ASCII text*, *symbolic link*) sont inclus.

2.5 Rectification d'une erreur

```
root@kevin-virtual-machine:/home/kevin# bash synchroniseur

== Bienvenue dans le système de synchronisation de fichiers ==

Entrer le 1er PATH de synchronisation: /home/userA/Repertoire_Partage

Entrer le 2eme PATH de synchronisation: /home/userA/Repertoire_Partage

Erreur: Vous avez entrer deux PATH identiques.

Le synchroniseur fonctionnera sous les chemins que vous specifiez :)
===== Merci de les entrer à nouveau. =====

Entrer le 1er PATH de synchronisation: /home/userA/Repertoire_Partage

Entrer le 2eme PATH de synchronisation: /home/userB/Repertoire_Partage
```

Si l'utilisateur entre deux chemins identiques au début, le terminal l'invite à entrer à nouveau.

Les messages d'erreur sont indiqués en **rouge**.

3. Implémentation du code

3.1 function *AddtoSynchro*

```
function AddtoSynchro {
    cd $1
    ls -R $1 | grep ./ | cut -d ':' -f 1|cut -d '.' -f 2- > /home/.cache_rep_temp
    while read -r line1
    do
        cd $line1
        ls -l | sed 's/ \+/ /g'| cut -d ' ' -f 9 > /home/.cache2
        sed -i '1d' /home/.cache2
        while read -r line2
        do

            echo $(pwd $line1/$line2) >> /home/.synchro
            a=$(ls -l $line1 | grep -w $line2 | sed 's/ \+/ /g'| cut -d ' ' -f 1,5,9)
            b=$(stat $line1/$line2 | awk 'NR==6 {print $0}'| cut -d ' ' -f 2,3)
            c=$(file -b $line2)
            sed -i '$s/$/ "'$a $b $c"' /' /home/.synchro

        done < /home/.cache2
        rm /home/.cache2
    done < /home/.cache_rep_temp
    rm /home/.cache_rep_temp
}
```

Cette fonction écrit le nom et les métadonnées(le type, les permissions, la taille et la date de modification) du fichier dans *.synchro*.

Ex: /home/userA/Repertoire Partage drwxr-xr-x 4096 world 2023-01-26 21:37:30.699484256

3.2 function *fiche_syn*

```

function fiche_syn {
touch /home/.synchro_temp
touch /home/.cache_compl

while read -r line;do  # Prend la même entrée dans le journal que ce chemin absolu et
la construit en tant que journal temporaire
    Temp_PATH=$(echo $line | cut -d " " -f 1)

    if [ ${Temp_PATH} = $1 ];then
        IsFile2=$(echo $line | cut -d " " -f 3)
        if [ ${IsFile2} -ne "4096" ];then
            echo $line >> /home/.synchro_temp
        fi

        elif [ ${Temp_PATH} = $2 ];then
            IsFile2=$(echo $line | cut -d " " -f 3)
            if [ ${IsFile2} -ne "4096" ];then
                echo $line >> /home/.synchro_temp  # /home/.synchro_temp Il est utilisé pour
stocker les enregistrements de fichiers liés au répertoire courant dans le journal
            fi
        fi
done</home/.synchro

touch /home/.cache_fiche
PATH2=${arr[1]}
PATH1=$1
CUT=$(echo ${PATH1}#${PATH2})
PathExist2=$(cat /home/.sous_reperatoire | grep -w $CUT)
if [ $? -eq 0 ];then
    ls -l $1 | sed 's/ \+/ /g' > /home/.cache_fiche      # cache_fiche Utilisé pour
stocker des informations sur tous les fichiers et dossiers du sous-répertoire actif
#eg: #####
## drwxr-xr-x 2 root root 4096 2023-1-13 22:28 eeeee  ##
## drwxr-xr-x 2 root root 4096 2023-1-13 22:28 asdasd ##
#####
sed -i 1d /home/.cache_fiche
fi
while read -r line;do
    IsFile=$(echo $line | sed 's/ \+/ /g'| cut -d ' ' -f 5)

    if [ ${IsFile} -ne "4096" ];then
        FILENAME=$(echo $line | cut -d " " -f 9)

        echo $FILENAME >> /home/.cache_compl
        TIME=$(stat $1/$FILENAME | awk 'NR==6 {print $0}'| cut -d ' ' -f 3)
        sed -i '$s/$/"$TIME"' / /home/.cache_compl  # Les informations du nouveau
fichier actuel sont stockées dans le journal
        #####/home/.cache_compl Stocke les informations de fichier dans le répertoire
actif  # eg: file1 12:57:12.488234726
    fi

done</home/.cache_fiche
rm /home/.cache_fiche

while read -r line  #Lire à partir de /home/.cache_compl
do
    Name=$(echo $line | cut -d " " -f 1)  # Obtenir une ligne à la fois avec le nom de
fichier
    temp=$(find $2 -name $Name | wc -L)  # Allez sur le chemin absolu opposé pour
savoir s'il y a ce nom de fichier

```

```

if [ "$temp " -eq "0" ];then # Si le chemin $2 ci-contre n'a pas ce nom de
fichier
    temp1=$(cat /home/.synchro_temp | grep -w $Name) # Il suffit d'aller dans le
journal temporaire pour trouver le fichier

    if [ $? -ne "0" ];then # Si le fichier est introuvable dans le journal
temporaire, cela signifie que le fichier a été recréé
        echo -e "\033[1;32m== Il y a un fichier <> $Name >> qui n'existe pas
==\033[0m" # ici on utilise la couleur vert claire
        mdftime2=$(stat $1 | awk 'NR==6 {print $0}'| cut -d ' ' -f 3) #Enregistrer
la date de modification du dossier avant l'opération
        cp -fdp $1/$Name $2 # Copiez ce fichier dans le chemin absolu de $2 ci-
contre
        touch -m -d "$(stat $1/$Name| awk 'NR==6 {print $0}'| cut -d ' ' -f 3)"
$2/$Name # Synchronisation de la date de modification des fichiers
        touch -m -d $mdftime2 $2 # Une fois l'opération terminée, restaurez la
date de modification du dossier externe

        echo $2 >>/home/.synchro # Les informations du nouveau fichier actuel sont
stockées dans le journal
        a=$(ls -l $2 | grep -w $Name | sed 's/ \+/ /g'| cut -d ' ' -f 1,5,9)
        b=$(stat $2/$Name | awk 'NR==6 {print $0}'| cut -d ' ' -f 2,3) # Les
informations du nouveau fichier actuel sont stockées dans le journal
        cd $2
        c=$(file -b $Name)
        sed -i '$s/$/ "'$a $b $c'" /' /home/.synchro # Les informations du
nouveau fichier actuel sont stockées dans le journal
        echo -e "\033[1;37m<< $Name >> a été copié par défaut (PATH:$2)\033[0m"
        echo
    -----
    flag=1

else # Si le fichier est trouvé dans le journal, cela signifie que le fichier
a effectivement été supprimé de $2, donc $1 doit également supprimer le fichier
correspondant pendant la synchronisation

    echo -e "\033[1;32m== Le fichier <> $Name >> a été supprimé par
l'utilisateur ==\033[0m"
    mdftime2=$(stat $1 | awk 'NR==6 {print $0}'| cut -d ' ' -f 3) #Enregistrer
la date de modification du dossier avant l'opération
    rm $1/$Name
    touch -m -d $mdftime2 $1 # Une fois l'opération terminée, restaurez la
date de modification du dossier
    temp3=$(cat /home/.synchro_temp | grep -w $Name | cut -d " " -f 3-6)
    sed -i "/${temp3}/d" /home/.synchro # Supprimer l'enregistrement de
fichier du journal
    echo -e "\033[1;37m<< $Name >> a été supprimé par défaut (PATH:$1)\033[0m"
    echo
    -----
    flag=1
fi

else # Cela correspond au cas où le même fichier se trouve dans le chemin opposé
$2
    StaTime=$(echo $line | cut -d " " -f 2)
    StaTime1=$(stat $1/$Name | awk 'NR==6 {print $0}'|cut -d ' ' -f 3)
    StaTime2=$(stat $2/$Name | awk 'NR==6 {print $0}'|cut -d ' ' -f 3)
    #echo 1---$Name,$StaTime,$StaTime1,$StaTime2
    if [ ${StaTime1} = ${StaTime2} ];then # Si le temps des deux côtés est égal,
sauvez-le directement et passez à la traversée suivante
        continue
    else
        echo "Le fichier $2 a été modifié depuis la dernière synchronisation.
        Il est recommandé de sauvegarder les modifications avant de continuer." >> /home/.synchro_error
        exit 1
    fi
fi

```

```

else
    if [ ${StaTime} = ${StaTime1} ];then # L'heure opposée est la même que
l'heure standard, ce qui signifie que la mienne est la dernière heure pour écraser
l'ancien fichier du côté opposé

        mdftime3=$(stat $2 | awk 'NR==6 {print $0}'| cut -d ' ' -f 3) #
Enregistrer la date de modification du dossier avant l'opération

        rm $2/$Name
        cp -fp $1/$Name $2
        touch -m -d "$(stat $1/$Name| awk 'NR==6 {print $0}'| cut -d ' ' -f 3)" $2/$Name
        touch -m -d $mdftime3 $2 # Une fois l'opération terminée, restaurez la
date de modification du dossier

temp3=$(cat /home/.synchro_temp | grep -w $Name | cut -d " " -f 3-6)
sed -i "/${temp3}/d" /home/.synchro
echo $2 >> /home/.synchro
a=$(ls -1 $2 | grep -w $Name | sed 's/ \+/ /g'| cut -d ' ' -f 1,5,9)

b=$(stat $2/$Name | awk 'NR==6 {print $0}'| cut -d ' ' -f 2,3)
cd $2
c=$(file -b $Name)
sed -i '$s/$/"$a $b $c"/' /home/.synchro

echo -e "\033[1;32m== Le fichier << $Name >> a été modifié ==\033[0m"
echo -e "\033[1;37mPATH: $2\033[0m"
echo

"_
#echo -e "\n"
flag=1
fi
fi
done < /home/.cache_compl
rm /home/.synchro_temp

EXIST=$(ls -a /home | grep -w .cache_compl | wc -L)
if [ ${EXIST} -ne 0 ];then
    rm /home/.cache_compl
fi
}

```

Cette fonction est utilisée pour synchroniser les fichiers dans un répertoire, la façon de déterminer le nouveau est de comparer avec la ligne en *.synchro*, si le nom du fichier en cours n'apparaît pas est ajouté. S'il est apparu, il doit être supprimé. Si la date de modification est différente, il est modifié, et la nouvelle heure écrase l'heure d'origine.

3.3 function *rep_syn*

```

function rep_syn {
PathExist=$(cat /home/.sous_reperatoire | grep -w $3)
if [ $? -eq "0" ];then
  cd $1$3
  ls -l | grep '^d'|sed 's/ \+/ /g'| cut -d ' ' -f 9 >/home/.cache3 # Répertoire de prise
uniquement (modifier)
  while read -r line3
  do
    IsFile=$(ls -l $1$3 | grep -w $line3 | sed 's/ \+/ /g'| cut -d ' ' -f 5)
    if [ ${IsFile} -eq "4096" ];then      # Si $line 3 est un dossier

      RES=$(cat /home/.sous_reperatoire_temp | grep -w $3)
      if [ $? -eq "0" ];then # S'il y a le même chemin du côté opposé
        cd $2$3
        temp1=$(find $2$3 -name $line3 | wc -L) # Déterminer s'il existe un dossier
$line 3 dans le répertoire opposé

        if [ $temp1 -eq 0 ];then # S'il n'y a pas de dossier en face
          temp2=$(cat /home/.synchro | cut -d " " -f 2-|grep -w $line3)
#####Accédez au journal pour voir s'il existe un répertoire temp3#####

          if [ $? -eq 0 ];then # Si trouvé
            temp2=$(echo $temp2 | cut -d " " -f 5) #temp2: Temps de
comparaison standard dans le journal
            cd $1$3 #切回原目录
            temp3=$(stat $line3 |awk 'NR==6 {print $0}'| cut -d ' ' -f 3 )
#Le temps d'obtenir le répertoire correspondant : temp3
            if [ ${temp2} = ${temp3} ];then # Il n'y a pas de
répertoire de ce type sur le côté opposé mais il y a un utilisateur actuel dans le
journal, donc il est également nécessaire de supprimer le répertoire de l'utilisateur
actuel

              echo -e "\033[1;32m== Le repertoire << $line3 >> a été
supprimé par l'utilisateur ==\033[0m"
              mdftime1=$(stat $1$3 | awk 'NR==6 {print $0}'| cut -d ' '
-f 3) # Enregistrer la date de modification du catalogue avant de modifier le contenu
du catalogue

              rm -rf $line3 # Supprimer le répertoire de
l'utilisateur actuel
              temp4=$(cat /home/.synchro | cut -d " " -f 2-| grep -w
$line3 | cut -d " " -f 2-5)

              sed -i "/${temp4}/d" /home/.synchro # Supprimer
l'enregistrement correspondant dans le journal

              touch -m -d $mdftime1 $1$3 # Remplacer la date de
retour
              cat -n /home/.sous_reperatoire | grep -w $line3 | sed
's/ \+/ /g'|cut -d " " -f 2 | cut -d "/" -f 1 >/home/.delete_temp
              echo -e "\033[1;37m<< $line3 >> a été supprimé par
defaut (PATH:$1$3)\033[0m"
              echo
"-----"
              flag=1
              while read -r line;do

```

```

        sed -i ${line}d /home/.sous_reperatoire

        done < /home/.delete_temp
        rm /home/.delete_temp
        fi
    else # Le dossier est introuvable dans le journal.Vous devez aller
du côté opposé pour créer le dossier
        echo -e "\033[1;32m== Il y a un repertoire <> $line3 >> qui
n'existe pas ==\033[0m"
        mdftime1=$(stat $2$3 | awk 'NR==6 {print $0}'| cut -d ' ' -f 3)
# Enregistrer la date de modification du catalogue avant de modifier le contenu du
catalogue
        mdftime2=$(stat $1$3/$line3 | awk 'NR==6 {print $0}'| cut -d ' '
-f 3) # Enregistrer la date de modification du répertoire avant de modifier le contenu

        cp -rfp $1$3/$line3 $2$3
AddtoSynchro $2$3/$line3

        touch -m -d $mdftime1 $2$3 # Remplacer la date de retour
        touch -m -d $mdftime2 $2$3/$line3 # Remplacer la date de
retour
        echo $2$3 >> /home/.synchro
a=$(ls -l $1$3 | grep -w $line3 | sed 's/ \+/ /g'| cut -d ' ' -
f 1,5,9)
        cd $1$3
c=$(file -b $line3)
b=$(stat $1$3/$line3 | awk 'NR==6 {print $0}'| cut -d ' ' -f
2,3)
        sed -i '$s/$/ '$a $b $c'' /' /home/.synchro
echo -e "\033[1;37m<> $line3 >> a été copié par défaut
(PATH:$2$3)\033[0m"
        echo
"-----
flag=1
fi
fi
else # Le côté opposé n'existe même pas Trouver ce répertoire $1$3 lui-même a été
supprimé par le côté opposé
    Res=$(cat /home/.synchro | cut -d " " -f 1 | grep -w $1$3)
    if [ $? -eq "0" ];then # Si le chemin lui-même peut être trouvé dans le
journal, cela signifie que le côté opposé a supprimé le chemin avant la synchronisation,
de sorte que le chemin lui-même doit être supprimé

        cd ..
        PATH=$(pwd)
        mdftime1=$(stat $PATH | awk 'NR==6 {print $0}'| cut -d ' ' -f 3) #
Enregistrer la date de modification précédente du catalogue avant de modifier le contenu
du répertoire

        rm -rf $1$3
        touch -m -d $mdftime1 $PATH # Remplacer la date de retour

        fi
        fi
    done</home/.cache3
    rm /home/.cache3
fi
fiche_syn $1$3 $2$3
}

```

Cette fonction est utilisée pour synchroniser les dossiers, et la façon de déterminer si la taille du dossier est “4096”. Si c'est “4096” , c'est un dossier.

Lors de l'exécution du programme, on utilisera d'abord **rep_syn**, puis on appellera **fiche_syn** dans chaque répertoire (sous-répertoire).

3.4 function **SYNCHRO**

```

function SYNCHRO {
    cd $1
    ls -R | grep ./ | cut -d ':' -f 1|cut -d '.' -f 2- > /home/.sous_reperatoire
    cd $2
    ls -R | grep ./ | cut -d ':' -f 1|cut -d '.' -f 2- > /home/.sous_reperatoire_temp
    printf "\033[?25l\033[1;32m[Progression]\033[0m \033[42m\033[1m %d%% %-9s\r\033[0m" 30
    sleep 0.05
    while read -r line;do
        rep_syn $1 $2 $line
    done</home/.sous_reperatoire
    printf "\033[?25l\033[1;32m[Progression]\033[0m \033[42m\033[1m %d%% %-15s\r\033[0m"
45
    sleep 0.05
    rm /home/.sous_reperatoire_temp
    rm /home/.sous_reperatoire
    cd $2
    ls -R | grep ./ | cut -d ':' -f 1|cut -d '.' -f 2- > /home/.sous_reperatoire
    cd $1
    ls -R | grep ./ | cut -d ':' -f 1|cut -d '.' -f 2- > /home/.sous_reperatoire_temp
    while read -r line;do
        rep_syn $2 $1 $line
    done</home/.sous_reperatoire
    printf "\033[?25l\033[1;32m[Progression]\033[0m \033[42m\033[1m %d%% %-28s\r\033[0m"
70
    sleep 0.05
    rm /home/.sous_reperatoire_temp
    rm /home/.sous_reperatoire
}

```

Cette fonction est utilisée pour parcourir et synchroniser les sous-dossiers dans les deux chemins de dossier utilisateur, et encapsulée dans la fonction pour rendre la logique du code plus claire et plus concise.

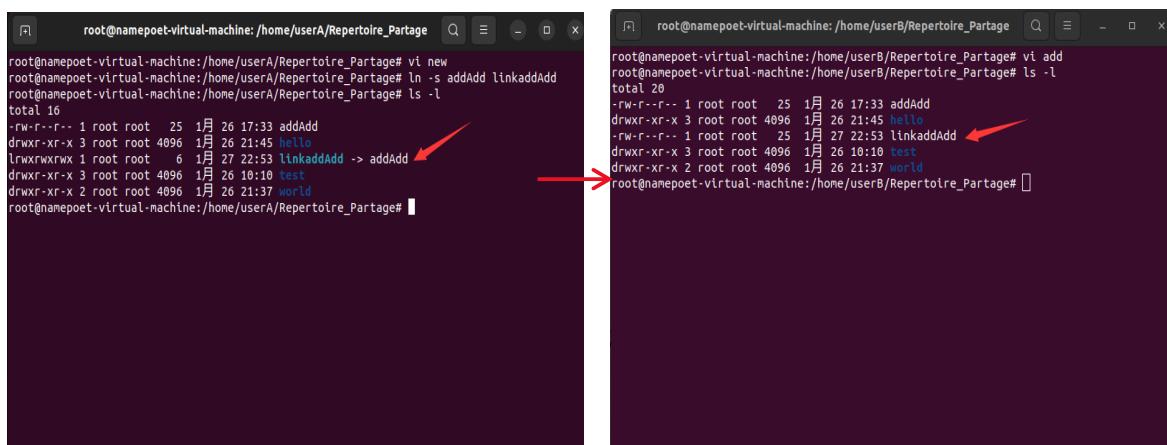
4. Division du travail

Prénom et Nom	Numéro	Responsable	Heures	Contribution
Chenjie LU	20124561	Discussion et conception du plan précédent, écriture du code de base (fonctions de synchronisation de fichiers et de dossiers), enregistrement du journal de travail et organisation du document	150h	55%
Yilin YANG	20124704	Discussion et conception du plan préliminaire, enregistrement du journal de travail, rédaction du rapport de projet, test de code, montage de la vidéo du projet et organisation du document	120h	45%

5. Amélioration et les autres

Nous avons également pensé et essayé la partie « EXTENSIONS », et ce que nous pouvons obtenir, c'est que le lien symbolique (soft link) est converti en un lien dur (hard link) après la synchronisation, mais le contenu du lien est exactement le même que le contenu du fichier lui-même.

Si vous copiez simplement le lien logiciel avec la commande « `cp -p` », le lien logiciel copié ne pourra pas afficher le contenu du fichier lié, nous avons donc pris la solution précédente.



```

root@namepoet-virtual-machine:/home/userA/Reperoire_Partage# vi new
root@namepoet-virtual-machine:/home/userA/Reperoire_Partage# ln -s addAdd linkaddAdd
root@namepoet-virtual-machine:/home/userA/Reperoire_Partage# ls -l
total 16
-rw-r--r-- 1 root root 25 1月 26 17:33 addAdd
drwxr-xr-x 3 root root 4096 1月 26 21:45 hello
lrwxrwxrwx 1 root root 6 1月 27 22:53 linkaddAdd -> addAdd
drwxr-xr-x 3 root root 4096 1月 26 10:10 test
drwxr-xr-x 2 root root 4096 1月 26 21:37 world
root@namepoet-virtual-machine:/home/userA/Reperoire_Partage# 

root@namepoet-virtual-machine:/home/userB/Reperoire_Partage# vi add
root@namepoet-virtual-machine:/home/userB/Reperoire_Partage# ls -l
total 20
-rw-r--r-- 1 root root 25 1月 26 17:33 addAdd
drwxr-xr-x 3 root root 4096 1月 26 21:45 hello
-rw-r--r-- 1 root root 25 1月 27 22:53 linkaddAdd
drwxr-xr-x 3 root root 4096 1月 26 10:10 test
drwxr-xr-x 2 root root 4096 1月 26 21:37 world
root@namepoet-virtual-machine:/home/userB/Reperoire_Partage# 

```

En ce qui concerne l'interface utilisateur, nous avons obtenu une interface conviviale et pratique en utilisant une variété de polices de couleurs et d'arrière-plans différents et en concevant une barre de progression pour afficher la progression de la synchronisation.

```

root@namepoet-virtual-machine: /home/namepoet/Desktop
===== La Synchronisation a commencé. Soyez patient SVP =====

==== Le fichier << new >> a été supprimé par l'utilisateur ===
<< new >> a été supprimé par défaut (PATH:/home/userA/Repertoire_Partage)
[Progression] 100%
===== Synchronisation est terminée =====

root@namepoet-virtual-machine:/home/namepoet/Desktop# bash synchroniseur_final

===== La Synchronisation a commencé. Soyez patient SVP =====

==== Le répertoire << WORLD >> a été supprimé par l'utilisateur ===
<< WORLD >> a été supprimé par défaut (PATH:/home/userA/Repertoire_Partage)
==== Il y a un répertoire << world >> qui n'existe pas ===
<< world >> a été copié par défaut (PATH: /home/userA/Repertoire_Partage)
[Progression] 100%
===== Synchronisation est terminée =====

root@namepoet-virtual-machine:/home/namepoet/Desktop# bash synchroniseur_final

===== La Synchronisation a commencé. Soyez patient SVP =====

===== Il y a un conflit =====
Raison: Les DEUX parties ont modifié ou créé << add >> avant la synchronisation

[1] PATH:/home/userA/Repertoire_Partage:
  1 ajouter
  2

[2] PATH:/home/userB/Repertoire_Partage:
  1 ajouter
  2
  3
  4 modifier

=====
Le fichier de quelle partie souhaitez-vous conserver? [1]/[2]2

```

6. Réflexion

Bien que j'ai bien compris les syntaxes du shell Linux en classe, dans le processus de réalisation du projet, j'ai découvert que ce projet ne nous oblige pas seulement à utiliser les connaissances de la classe, mais nous oblige également à élargir nos connaissances après la classe pour réaliser toutes les fonctions du synchroniseur de fichiers. De plus, la conception de la structure du code et la discussion avec un autre membre de l'équipe sont également essentielles.

(Chenjie LU - 20124561)

La réalisation de ce projet Synchroniseur m'a permis de mieux comprendre et de prendre conscience de nombreuses commandes de console Linux enseignées en classe. En même temps, j'ai appris à utiliser ces commandes correctement. La combinaison de la théorie et de la pratique m'a fait sentir que je pouvais apprendre beaucoup dans le cours LO03.

(Yilin YANG - 20124704)

7. Référence

- [1]. The Linux Command Line, 2nd Edition, William Shotts, 2019, ISBN 978-1593279523
- [2]. Wicked Cool Shell Scripts, 2nd Edition, Dave Taylor and Brandon Perry, 2017, ISBN 978-1-59327-602-7
- [3]. <https://www.csdn.net>
- [4]. <https://www.linuxcommand.org/>

Annexe (code de projet complet)

```

01.#!/bin/bash
02.function AddtoSynchro {
03.    cd $1
04.    ls -R $1 | grep ./ | cut -d ':' -f 1|cut -d '.' -f 2- > /home/.cache_rep_temp
05.    while read -r line1
06.    do
07.        cd $line1
08.        ls -l | sed 's/ \+/ /g'| cut -d ' ' -f 9 > /home/.cache2
09.        sed -i '1d' /home/.cache2
10.        while read -r line2
11.        do
12.
13.            echo $(pwd $line1/$line2) >> /home/.synchro
14.            a=$(ls -l $line1 | grep -w $line2 | sed 's/ \+/ /g'| cut -d ' ' -f 1,5,9)
15.            b=$(stat $line1/$line2 | awk 'NR==6 {print $0}'| cut -d ' ' -f 2,3)
16.            c=$(file -b $line2)
17.            sed -i '$s/$/"$a $b $c"/' /home/.synchro
18.
19.        done < /home/.cache2
20.        rm /home/.cache2
21.    done < /home/.cache_rep_temp
22.    rm /home/.cache_rep_temp
23.}
24.
25.function fiche_syn {
26.    touch /home/.synchro_temp
27.    touch /home/.cache_compl
28.
29.    while read -r line;do
30.        Temp_PATH=$(echo $line | cut -d " " -f 1)
31.
32.        if [ ${Temp_PATH} = $1 ];then
33.            IsFile2=$(echo $line | cut -d " " -f 3)
34.            if [ ${IsFile2} -ne "4096" ];then
35.                echo $line >> /home/.synchro_temp
36.            fi
37.
38.        elif [ ${Temp_PATH} = $2 ];then
39.            IsFile2=$(echo $line | cut -d " " -f 3)
40.            if [ ${IsFile2} -ne "4096" ];then
41.                echo $line >> /home/.synchro_temp
42.            fi
43.        fi
44.    done</home/.synchro
45.
46.    touch /home/.cache_fiche
47.    PATH2=${arr[1]}
48.    PATH1=$1
49.    CUT=$(echo ${PATH1}#${PATH2})
50.    PathExist2=$(cat /home/.sous_reperatoire | grep -w $CUT)
51.    if [ $? -eq 0 ];then
52.        ls -l $1 | sed 's/ \+/ /g' > /home/.cache_fiche
53.
54.        sed -i 1d /home/.cache_fiche
55.    fi
56.    while read -r line;do
57.        IsFile=$(echo $line | sed 's/ \+/ /g'| cut -d ' ' -f 5)
58.
59.        if [ ${IsFile} -ne "4096" ];then
60.            FILENAME=$(echo $line | cut -d " " -f 9)
61.
62.            echo $FILENAME >> /home/.cache_compl
63.            TIME=$(stat $1/$FILENAME | awk 'NR==6 {print $0}'| cut -d ' ' -f 3)
64.            sed -i '$s/$/"$TIME"/' /home/.cache_compl
65.        fi
66.
67.    done</home/.cache_fiche
68.    rm /home/.cache_fiche
69.
70.    while read -r line
71.    do
72.        Name=$(echo $line | cut -d " " -f 1)
73.        temp=$(find $2 -name $Name | wc -L)
74.        if [ "$temp" -eq "0" ];then
75.            temp1=$(cat /home/.synchro_temp | grep -w $Name)
76.            if [ $? -ne "0" ];then
77.                echo -e "\033[1;32m== Il y a un fichier << $Name >> qui n'existe pas ==\033[0m"
78.                mdftime2=$(stat $1 | awk 'NR==6 {print $0}'| cut -d ' ' -f 3)
79.                cp -fp $1/$Name $2
80.                touch -m -d "$(stat $1/$Name | awk 'NR==6 {print $0}'| cut -d ' ' -f 3)" $2/$Name
81.                touch -m -d $mdftime2 $2
82.                echo $2 >>/home/.synchro
83.                a=$(ls -l $2 | grep -w $Name | sed 's/ \+/ /g'| cut -d ' ' -f 1,5,9)

```

```

84.         b=$(stat $2/$Name | awk 'NR==6 {print $0}'| cut -d ' ' -f 2,3)
85.         cd $2
86.         c=$(file -b $Name)
87.         sed -i '$s/$/ "'$a $b $c'" /' /home/.synchro
88.         echo -e "\033[1;37m<< $Name >> a été copié par défaut (PATH:$2)\033[0m"
89.         echo "-----"
90.         flag=1
91.
92.     else
93.         echo -e "\033[1;32m==> Le fichier << $Name >> a été supprimé par l'utilisateur ==\033[0m"
94.         mdftime2=$(stat $1 | awk 'NR==6 {print $0}'| cut -d ' ' -f 3)
95.         #Time =$(stat $1/$Name | awk 'NR==6 {print $0}'| cut -d ' ' -f 3)
96.         rm $1/$Name
97.         touch -m -d $mdftime2 $1
98.         temp3=$(cat /home/.synchro_temp | grep -w $Name | cut -d " " -f 3-6)
99.         sed -i "/${temp3}/d" /home/.synchro
100.        echo -e "\033[1;37m<< $Name >> a été supprimé par défaut (PATH:$1)\033[0m"
101.        echo "-----"
102.        flag=1
103.    fi
104.
105. else
106.
107.     StaTime=$(echo $line | cut -d " " -f 2)
108.     StaTime1=$(stat $1/$Name | awk 'NR==6 {print $0}'|cut -d ' ' -f 3)
109.     StaTime2=$(stat $2/$Name | awk 'NR==6 {print $0}'|cut -d ' ' -f 3)
110.     echo 1--$Name, ${StaTime}, ${StaTime1}, ${StaTime2}
111.     if [ ${StaTime1} = ${StaTime2} ];then
112.         continue
113.     else
114.         if [ ${StaTime} = ${StaTime1} ];then
115.             mdftime3=$(stat $2 | awk 'NR==6 {print $0}'| cut -d ' ' -f 3)
116.             rm $2/$Name
117.             cp -fp $1/$Name $2
118.             touch -m -d "$(stat $1/$Name| awk 'NR==6 {print $0}'| cut -d ' ' -f 3)" $2/$Name
119.             touch -m -d $mdftime3 $2
120.             temp3=$(cat /home/.synchro_temp | grep -w $Name | cut -d " " -f 3-6)
121.             sed -i "/${temp3}/d" /home/.synchro
122.             echo $2 >> /home/.synchro
123.             a=$(ls -l $2 | grep -w $Name | sed 's/ \+/ /g'| cut -d ' ' -f 1,5,9)
124.             b=$(stat $2/$Name | awk 'NR==6 {print $0}'| cut -d ' ' -f 2,3)
125.             cd $2
126.             c=$(file -b $Name)
127.             sed -i '$s/$/ "'$a $b $c'" /' /home/.synchro
128.             echo -e "\033[1;32m==> Le fichier << $Name >> a été modifié ==\033[0m"
129.             echo -e "\033[1;37mPATH: $2\033[0m"
130.             echo "-----"
131.             flag=1
132.         fi
133.     fi
134. done < /home/.cache_compl
135. rm /home/.synchro_temp
136.
137. EXIST=$(ls -a /home | grep -w .cache_compl | wc -L)
138. if [ ${EXIST} -ne 0 ];then
139.     rm /home/.cache_compl
140. fi
141.
142. }
143.
144. function rep_syn {
145. PathExist=$(cat /home/.sous_repertoire | grep -w $3)
146. if [ $? -eq "0" ];then
147.     cd $1$3
148.     ls -l | grep '^d'|sed 's/ \+/ /g'| cut -d ' ' -f 9 >/home/.cache3
149.     while read -r line3
150.     do
151.         IsFile=$(ls -l $1$3 | grep -w $line3 | sed 's/ \+/ /g'| cut -d ' ' -f 5)
152.         if [ ${IsFile} -eq "4096" ];then
153.
154.             RES=$(cat /home/.sous_repertoire_temp | grep -w $3)
155.             if [ $? -eq "0" ];then
156.                 cd $2$3
157.                 temp1=$(find $2$3 -name $line3 | wc -L)
158.                 if [ $temp1 -eq 0 ];then
159.                     temp2=$(cat /home/.synchro | cut -d " " -f 2-|grep -w $line3)
160.                     if [ $? -eq 0 ];then #如果找到
161.                         temp2=$(echo $temp2 | cut -d " " -f 5)
162.                         cd $1$3
163.                         temp3=$(stat $line3 |awk 'NR==6 {print $0}'| cut -d ' ' -f 3 )
164.                         if [ ${temp2} = ${temp3} ];then
165.                             echo -
e "\033[1;32m==> Le repertoire << $line3 >> a été supprimé par l'utilisateur ==\033[0m"

```

```

166.           mdftime1=$(stat $1$3 | awk 'NR==6 {print $0}'| cut -d ' ' -f 3)
167.           rm -rf $line3
168.           temp4=$(cat /home/.synchro | cut -d " " -f 2-| grep -w $line3 | cut -d " " -f 2-
5)
169.           sed -i "/${temp4}/d" /home/.synchro
170.           touch -m -d $mdftime1 $1$3
171.           cat -n /home/.sous_repertoire | grep -w $line3 | sed 's/ \+/ /g'|cut -d " " -f 2 | cut -
d "/" -f 1 > /home/.delete_temp
172.           echo -e "\033[1;37m<< $line3 >> a été supprimé par défaut (PATH:$1$3)\033[0m"
173.           echo "_____"
174.           flag1
175.           while read -r line;do
176.               sed -i $(($line))d /home/.sous_repertoire
177.
178.           done < /home/.delete_temp
179.           rm /home/.delete_temp
180.           fi
181.       else
182.           echo -
e "\033[1;32m== Il y a un repertoire << $line3 >> qui n'existe pas ==\033[0m"
183.           mdftime1=$(stat $2$3 | awk 'NR==6 {print $0}'| cut -d ' ' -f 3)
184.           mdftime2=$(stat $1$3/$line3 | awk 'NR==6 {print $0}'| cut -d ' ' -f 3)
185.           cp -rfp $1$3/$line3 $2$3
186.           AddtoSynchro $2$3/$line3
187.           touch -m -d $mdftime1 $2$3
188.           touch -m -d $mdftime2 $2$3/$line3
189.           echo $2$3 >> /home/.synchro
190.           a=$(ls -l $1$3 | grep -w $line3 | sed 's/ \+/ /g' | cut -d ' ' -f 1,5,9)
191.           b=$(stat $1$3/$line3 | awk 'NR==6 {print $0}'| cut -d ' ' -f 2,3)
192.           cd $1$3
193.           c=$(file -b $line3)
194.           sed -i '$$/s/"$a $b $c"/' /home/.synchro
195.           echo -e "\033[1;37m<< $line3 >> a été copié par défaut (PATH:$2$3)\033[0m"
196.           echo "_____"
197.           flag1
198.       fi
199.   fi
200. else
201.     Res=$(cat /home/.synchro | cut -d " " -f 1 | grep -w $1$3)
202.     if [ $? -eq "0" ];then
203.       cd ..
204.       PATH=$(pwd)
205.       mdftime1=$(stat $PATH | awk 'NR==6 {print $0}'| cut -d ' ' -f 3)
206.       rm -rf $1$3
207.       touch -m -d $mdftime1 $PATH
208.     fi
209.   fi
210.   fi
211.   done</home/.cache3
212.   rm /home/.cache3
213. fi
214. fiche_syn $1$3 $2$3
215. }
216.
217. function SYNCHRO {
218.   cd $1
219.   ls -R | grep ./ | cut -d ':' -f 1|cut -d '.' -f 2- > /home/.sous_repertoire
220.   cd $2
221.   ls -R | grep ./ | cut -d ':' -f 1|cut -d '.' -f 2- > /home/.sous_repertoire_temp
222.   printf "\033[?251\033[1;32m[Progression]\033[0m \033[42m\033[1m %d%% %s\033[0m" 30
223.   sleep 0.05
224.   while read -r line;do
225.     rep_syn $1 $2 $line
226.     done</home/.sous_repertoire
227.     printf "\033[?251\033[1;32m[Progression]\033[0m \033[42m\033[1m %d%% %s\033[0m" 45
228.     sleep 0.05
229.     rm /home/.sous_repertoire_temp
230.     rm /home/.sous_repertoire
231.   cd $2
232.   ls -R | grep ./ | cut -d ':' -f 1|cut -d '.' -f 2- > /home/.sous_repertoire
233.   cd $1
234.   ls -R | grep ./ | cut -d ':' -f 1|cut -d '.' -f 2- > /home/.sous_repertoire_temp
235.   while read -r line;do
236.     rep_syn $2 $1 $line
237.     done</home/.sous_repertoire
238.     printf "\033[?251\033[1;32m[Progression]\033[0m \033[42m\033[1m %d%% %s\033[0m" 70
239.     sleep 0.05
240.     rm /home/.sous_repertoire_temp
241.     rm /home/.sous_repertoire
242.   }
243.
244. cd /home
str='Progression:'

```

```

246.
247. flag=0
248. exist1=$(find /home -name .cache_syn | wc -L)
249. if [ "$exist1" -eq "0" ];then
250. touch .cache_syn
251. echo -e "\n"
252. echo -e "\033[1;32m== Bienvenue dans le système de synchronisation de fichiers ==\033[0m"
253. while :;do
254.     echo -e "\n"
255.     read -p "Entrer le 1er PATH de synchronisation: " PATH1
256.     echo -e "\n"
257.     read -p "Entrer le 2ème PATH de synchronisation: " PATH2
258.     if [ ${PATH1} = ${PATH2} ];then
259.         echo -e "\n"
260.         echo -e "\033[1;31m Erreur: Vous avez entré deux PATH identiques.\033[0m"
261.         echo -e "\n"
262.         echo -e "\033[37m Le synchroniseur fonctionnera sous les chemins que vous specifiez :) \033[0m"
263.         echo -e "\033[1;32m ===== Merci de les entrer a nouveau. =====\033[0m"
264.     else
265.         break
266.     fi
267. done
268. echo $PATH1 >/home/.cache_syn
269. echo $PATH2 >>/home/.cache_syn
270. fi
271.
272. arr=()
273. Num=0
274.
275. while read -r line
276. do
277.     arr[$Num]=$line
278.     Num=$((expr $Num+1))
279.     echo ${arr[$Num]}
280. done < /home/.cache_syn
281.
282. exist2=$( find /home -name .synchro | wc -L)
283. if [ "$exist2" -eq "0" ];then
284.     touch .synchro
285. fi
286.
287. echo -e "\033[1;46;30m===== La Synchronisation a commencé. Soyez patient SVP =====\033[0m"
288. echo -e "\n"
289.
290. printf "\033[?25l\033[1;32m[Progression]\033[0m \033[42m\033[1m %d%% \r\033[0m" 0
291. sleep 0.05
292.
293. for num in {0..1}
294. do
295.     if [ 1 -eq $num ];then
296.         temp=${arr[num]}
297.         arr[num]="${arr[num-1]}"
298.         arr[num-1]=$temp
299.     fi
300.
301.     cd ${arr[0]}
302.     for i in $(ls);do
303.         temp=$(ls ${arr[1]} | grep -w $i | wc -L)
304.         if [ "$temp" -eq "0" ];then
305.             touch /home/.FIND_CACHE
306.             cat /home/.synchro | grep -w $i > /home/.FIND_CACHE
307.             if [ $? -eq 0 ];then
308.                 while read -r LINE;do
309.                     IsInRoot=$(echo $LINE | cut -d ' ' -f 1)
310.                     if [ ${IsInRoot} = ${arr[1]} ];then
311.                         Absent=2
312.                         break
313.                     elif [ ${IsInRoot} = ${arr[0]} ];then
314.                         Absent=2
315.                         break
316.                     else Absent=1
317.                         fi
318.                 done < /home/.FIND_CACHE
319.             else
320.                 Absent=1
321.             fi
322.             rm /home/.FIND_CACHE
323.
324.             if [ $Absent -eq 1 ];then
325.                 Isfile=$(ls -l ${arr[0]} | grep -w $i | sed 's/ \+/ /g' | cut -d ' ' -f 5)
326.                 if [ ${Isfile} -eq "4096" ];then
327.                     echo -e "\033[1;32m== Il y a un repertoire << $i >> qui n'existe pas ==\033[0m"
328.                     cp -rfp ${arr[0]}/$i ${arr[1]}

```

```

329.           AddtoSynchro ${arr[1]}/$i
330.       else
331.           echo -e "\033[1;32m== Il y a un fichier << $i >> qui n'existe pas ==\033[0m"
332.
333.           cp -fp ${arr[0]}/$i ${arr[1]}
334.       fi
335.       touch -m -d "$(stat ${arr[0]}/$i | awk 'NR==6 {print $0}' | cut -d ' ' -f 3)" ${arr[1]}/$i
336.       a=$(ls -l ${arr[0]} | grep -w $i | sed 's/ \+/ /g' | cut -d ' ' -f 1,5,9)
337.       b=$(stat ${arr[0]}/$i | awk 'NR==6 {print $0}' | cut -d ' ' -f 2,3)
338.       cd ${arr[0]}
339.       c=$(file -b $i)
340.       echo ${arr[1]} >> /home/.synchro
341.       sed -i '$$/s/\$a \$b \$c/' /home/.synchro
342.       echo -e "\033[1;37m << $i >> a été copié par défaut (PATH: ${arr[1]}) \033[0m"
343.       echo "
344. #echo -e "\n"
345. flag1
346. elif [ $Absent -eq 2 ];then
347.     IsFile=$(ls -l ${arr[0]} | grep -w $i | sed 's/ \+/ /g' | cut -d ' ' -f 5)
348.     if [ ${IsFile} -eq "4096" ];then
349.         echo -e "\033[1;32m== Le répertoire << $i >> a été supprimé par l'utilisateur ==\033[0m"
350.         touch /home/.DELETE_CACHE
351.         ls -R ${arr[0]}/$i | grep ./ | cut -d ':' -f 1|cut -d '.' -f 2- >/home/.DELETE_CACHE
352.         while read -r KEY;do
353.             res1=$(echo ${arr[0]}${KEY##${arr[0]}})
354.             res2=$(echo ${arr[1]}${KEY##${arr[0]}})
355.             touch /home/TEMP
356.             cat /home/.synchro | grep -w $res1 > /home/TEMP
357.             while read -r line;do
358.                 DEL=$(echo $line | cut -d " " -f 4-6)
359.                 sed -i "/${DEL}/d" /home/.synchro
360.             done</home/TEMP
361.             rm /home/TEMP
362.
363.             touch /home/TEMP
364.             cat /home/.synchro | grep -w $res2 > /home/TEMP
365.             while read -r line;do
366.                 DEL=$(echo $line | cut -d " " -f 4-6)
367.                 sed -i "/${DEL}/d" /home/.synchro
368.             done</home/TEMP
369.             rm /home/TEMP
370.
371.             done< /home/.DELETE_CACHE
372.             rm /home/.DELETE_CACHE
373.         else
374.             echo -e "\033[1;32m== Le fichier << $i >> a été supprimé par l'utilisateur ==\033[0m"
375.         fi
376.
377.         touch /home/.SEARCH_CACHE
378.         cat /home/.synchro | grep -w $i > /home/.SEARCH_CACHE
379.
380.         while read -r S_Line;do
381.             IsInRoot=$(echo $S_Line | cut -d ' ' -f 1)
382.             if [ ${IsInRoot} = ${arr[0]} ];then
383.                 Delete_Key=$(echo $S_Line | cut -d " " -f 6)
384.                 sed -i "/${Delete_Key}/d" /home/.synchro
385.                 rm -rf ${arr[0]}/$i
386.             elif [ ${IsInRoot} = ${arr[1]} ];then
387.                 Delete_Key=$(echo $S_Line | cut -d " " -f 6)
388.                 sed -i "/${Delete_Key}/d" /home/.synchro
389.                 rm -rf ${arr[0]}/$i
390.             fi
391.             done< /home/.SEARCH_CACHE
392.             rm /home/.SEARCH_CACHE
393.             echo -e "\033[1;37m << $i >> a été supprimé par défaut (PATH:${arr[0]}) \033[0m"
394.             echo "
395. flag1
396.         fi
397.     else
398.         touch /home/.SEARCH_CACHE
399.         cat /home/.synchro | grep -w $i > /home/.SEARCH_CACHE
400.         while read -r line;do
401.             IsInRoot=$(echo $line | cut -d ' ' -f 1)
402.             if [ ${IsInRoot} = ${arr[0]} ];then
403.                 FIND_SYN=1
404.                 break
405.             elif [ ${IsInRoot} = ${arr[1]} ];then
406.                 FIND_SYN=1
407.                 break
408.             else
409.                 FIND_SYN=0
410.             fi

```

```

411. done</home/.SEARCH_CACHE
412. count=$(cat /home/.SEARCH_CACHE | wc -L)
413. if [ $count -eq 0 ];then
414.     FIND_SYN=0
415. fi
416. rm /home/.SEARCH_CACHE
417. StatTime1=$(stat ${arr[0]}/$i | awk 'NR==6 {print $0}'| cut -d ' ' -f 3)
418. StatTime2=$(stat ${arr[1]}/$i | awk 'NR==6 {print $0}'| cut -d ' ' -f 3)
419. if [ $FIND_SYN -eq 0 ];then
420.     if [ ${StatTime1} = ${StatTime2} ];then
421.         echo ${arr[0]}>>/home/.synchro
422.         a=$(ls -l ${arr[0]} | grep -w $i | sed 's/ \+/ /g'| cut -d ' ' -f 1,5,9)
423.         b=$(stat ${arr[0]}/$i | awk 'NR==6 {print $0}'| cut -d ' ' -f 2,3)
424.         cd ${arr[0]}
425.         c=$(file -b $i)
426.         sed -i '$s/$/"$a $b $c"/' /home/.synchro
427.         flag=1
428.     fi
429. fi
430. if [ ${StatTime1} != ${StatTime2} ];then
431.
432.     IsFileA=$(ls -l ${arr[0]} | grep -w $i | sed 's/ \+/ /g'| cut -d ' ' -f 5)
433.     IsFileB=$(ls -l ${arr[1]} | grep -w $i | sed 's/ \+/ /g'| cut -d ' ' -f 5)
434.     if [ $IsFileA -ne 4096 ];then
435.         if [ $IsFileB -ne 4096 ];then
436.             echo -
437.             e "\033[1;41;30m ===== Il y a un conflit ===== \033[0m"
438.             echo -
439.             e "\033[1;31m Raison: Les DEUX parties ont modifié ou créé <> $i <> avant la synchronisation \033[0m"
440.             echo -e "\n"
441.             echo -e "\033[1;33m [1] PATH:${arr[0]}: \033[0m"
442.             cat -n ${arr[0]}/$i
443.             echo -e "\n"
444.             echo -e "\033[1;33m [2] PATH:${arr[1]}: \033[0m"
445.             cat -n ${arr[1]}/$i
446.             echo -e "\n"
447.             echo -
448.             e "\033[1;41;30m ===== \033[0m"
449.             flag=1
450.             while :;do
451.                 echo -e "\n"
452.                 read -p "Le fichier de quelle partie souhaitez-vous conserver? [1]/[2]" RES
453.                 echo -e "\n"
454.                 if [ $RES -eq 1 ];then
455.                     echo -e "\n"
456.                     echo -e "\033[1;32m== <> $i <> de PATH:${arr[0]} a été conservé ==\033[0m"
457.                     echo -e "\n"
458.                     if [ $FIND_SYN -eq 1 ];then
459.                         rm ${arr[1]}/$i
460.                         sed -i '/$i/d' /home/.synchro
461.                     fi
462.                     echo ${arr[0]}>>/home/.synchro
463.                     a=$(ls -l ${arr[0]} | grep -w $i | sed 's/ \+/ /g'| cut -d ' ' -f 1,5,9)
464.                     b=$(stat ${arr[0]}/$i | awk 'NR==6 {print $0}'| cut -d ' ' -f 2,3)
465.                     cd ${arr[0]}
466.                     c=$(file -b $i)
467.                     sed -i '$s/$/"$a $b $c"/' /home/.synchro
468.                     cp -rfp ${arr[0]}/$i ${arr[1]}
469.                     break
470.                 elif [ $RES -eq 2 ];then
471.                     echo -e "\n"
472.                     echo -e "\033[1;32m== <> $i <> de PATH:${arr[1]} a été conservé ==\033[0m"
473.                     echo -e "\n"
474.                     if [ $FIND_SYN -eq 1 ];then
475.                         rm ${arr[0]}/$i
476.                         sed -i '/$i/d' /home/.synchro
477.                     fi
478.                     echo ${arr[1]}>>/home/.synchro
479.                     a=$(ls -l ${arr[1]} | grep -w $i | sed 's/ \+/ /g'| cut -d ' ' -f 1,5,9)
480.                     b=$(stat ${arr[1]}/$i | awk 'NR==6 {print $0}'| cut -d ' ' -f 2,3)
481.                     cd ${arr[1]}
482.                     c=$(file -b $i)
483.                     sed -i '$s/$/"$a $b $c"/' /home/.synchro
484.                     cp -rfp ${arr[1]}/$i ${arr[0]}
485.                     break
486.                 else
487.                     echo -e "\n"
488.                     echo -e "\033[1;31mChoisir entre les deux fichiers, s'il vous plaît. [1]/[2]\033[0m"
489.                 fi
490.             done
491.         fi
492.     fi

```

```

491.         fi
492.     fi
493.     Absent=0
494.     done
495. done
496.
497. printf "\033[?25l\033[1;32m[Progression]\033[0m \033[42m\033[1m %d%% %-4s\r\033[0m" 15
498. sleep 0.05
499.
500. SYNCHRO ${arr[0]} ${arr[1]}
501.
502. printf "\033[?25l\033[1;32m[Progression]\033[0m \033[42m\033[1m %d%% %-40s\r\033[0m" 100
503. sleep 0.05
504. printf "\033[?25h"
505. if [ $flag -eq 0 ];then
506.   printf "\n"
507.   printf "\n"
508.   echo -e "\033[1;32m ===== Aucun fichier&Repertoire a synchronisé ===== \033[0m"
509.
510.
511. echo -e "\n"
512. echo -e "\033[1;46;30m===== Synchronisation est terminée ======\033[0m"
513. echo -e "\n"
514.

```