TP1 - PARTIE 4

IV. Synchronisation de fichiers en bash

On souhaite développer une solution de synchronisation de fichiers.

Dans cette partie nous aurons besoin de 3 conteneurs C1, C2 et C3, qui seront créés avec les scripts précédents.

Table des matières

IV. **Synchronisation** de fichiers en bash 1

1. Création des trois conteneurs 2
2. Synchronisation 2

a. Créer un script permettant l'**ajout de répertoires ou de fichiers** à la liste des sauvegardes / synchronisations ; 2

b. Créer un script permettant la **synchronisation d'un fichier**, d'un répertoire ou de l'ensemble des éléments enregistrés pour être synchronisés. 4

c. Créer un script depuis la machine hôte pour **piloter** l'exemple d'exécution. 9

d. Une fois les scripts écrits effectuer un exemple d'exécution 11

1. Création des trois conteneurs

Une fois dans le dossier où se trouve mes scripts fait dans la *Partie III.2* rentrer les commandes suivantes afin de créer 3 conteneurs C1, C2 et C3 :

```
sudo ./creat_container.sh ubuntu C1 lxcbr0
sudo ./creat_container.sh ubuntu C2 lxcbr0
sudo ./creat_container.sh ubuntu C3 lxcbr0
```

Puis on les allume :

```
sudo ./start_container.sh C*
```

```
NAME STATE   AUTOSTART GROUPS IPV4       IPV6 UNPRIVILEGED
C1   RUNNING 0         -      10.0.3.2   -    false
C2   RUNNING 0         -      10.0.3.111 -    false
C3   RUNNING 0         -      10.0.3.244 -    false
```

J'ai fait le choix de laisser le bridge par défaut (`lxcbr0`). Il est tout de même recommandé de tester les connections entre les différents conteneurs avec la commande « `ping` » pour être certain que les conteneurs puissent communiquer entre eux et donc se synchroniser.

2. Synchronisation

La synchronisation sera dirigée par un utilisateur, qui lancera explicitement un script, et non par un processus en tâche de fond.

On souhaite mettre en place une solution de synchronisation.

C1 jouera le rôle de serveur de sauvegarde. Sur **C2 et C3 on définira des répertoires à synchroniser**. Lors d'une phase de synchronisation, chacun des fichiers à synchroniser sera comparé avec sa version sur le serveur. Si les deux fichiers sont identiques, rien ne sera effectué. Si les deux fichiers ne sont pas identiques, le plus récent remplacera le plus ancien, et ce quelle que soit la localisation du plus ancien, sur C1, C2 ou C3.

a. Créer un script permettant l'ajout de répertoires ou de fichiers à la liste des sauvegardes / synchronisations ;

1. J'ai créé un premier script de trois lignes : `crea_fichier_synch.sh`, qui permet de créer le fichier « SYNCH », fichier contenant les chemins des répertoires et des fichiers devant être synchronisés (optionnel, sera créé à la première écriture sinon).

```
#! /bin/bash
if [ $# -ne 1 ]; then
    echo "Il manque le nom du conteneur où le fichier SYNCH.txt doit être
créé."
else
    nom=$1
    lxc-attach -n $nom -- mkdir -p Partie_4
    lxc-attach -n $nom -- touch Partie_4/SYNCH.txt
    echo Le fichier de synchronisation a été créé.
fi
```

2. J'ai créé un script, `ajout_dossier_fichier.sh`, permettant de rajouter les chemins des fichiers et des dossiers à synchroniser. J'écris également la date (minute : heure : jour : mois) qui permet de garder la date de la dernière synchronisation.

```
#! /bin/bash
if [ $# -ne 2 ]
then
    echo "Le nombre d argument n est pas bon -- [1 : nom du conteneur ] [2 : nom
du fichier ou du dossier (avec son PATH après Partie_4 en entier)]"
    exit 1
else
    conteneur=$1
    fichier_dossier=$2
    date=$(date +'%M:%H:%d:%m')

    # On vérifie si le fichier ou le dossier existe
    if [ -e /var/lib/lxc/$conteneur/rootfs/Partie_4/$fichier_dossier ]; then

        #si oui on rajoute a la liste du conteneur
        deja_dedans=false
        while IFS= read -r line; do
        last_part=$(echo "$line" | awk '{print $NF}')
        if [ "$fichier_dossier" = "$last_part" ]; then
            deja_dedans=true
            break
        fi
        done < /var/lib/lxc/$conteneur/rootfs/Partie_4/SYNCH.txt
        if [ "$deja_dedans" = "false" ]; then
            lxc-attach -n "$conteneur" -- bash -c "echo '$date $fichier_dossier'
>> /Partie_4/SYNCH.txt"
        fi


        #ainsi que dans la liste du serveur
        deja_dedans_C1=false
        while IFS= read -r line; do
        last_part=$(echo "$line" | awk '{print $NF}')
        if [ "$fichier_dossier" = "$last_part" ]; then
            deja_dedans_C1=true
            break
        fi
        done < /var/lib/lxc/C1/rootfs/Partie_4/SYNCH.txt
        if [ "$deja_dedans_C1" = "false" ]; then
            lxc-attach -n C1 -- bash -c "echo '$date $fichier_dossier' >>
/Partie_4/SYNCH.txt"
        fi

        echo "$fichier_dossier a bien été noté comme dossier/fichier à
synchroniser."

        #on copie le fichier sur le serveur
        if [ "$deja_dedans_C1" = "false" ]; then
            cp -r /var/lib/lxc/$conteneur/rootfs/Partie_4/$fichier_dossier
/var/lib/lxc/C1/rootfs/Partie_4/SYNCH/$fichier_dossier
        fi
    else
        echo "$fichier_dossier n'existe pas, impossible de l'ajouter au fichier
de synchronisation."
```

```
    fi
fi
```

b. Créer un script permettant la synchronisation d'un fichier, d'un répertoire ou de l'ensemble des éléments enregistrés pour être synchronisés.

Informations supplémentaires : À chaque exécution d'une synchronisation :

- Le script affichera l'état de la synchronisation sur C2 et C3
- Le script sauvegardera dans un fichier les opérations réalisées sur C1.

- Le script `fichier_serveur.sh` créé un fichier `operations.txt` qui permet de sauvegarder les opérations réalisées (optionnel, sera créé à la première écriture sinon).

```
#! /bin/bash
lxc-attach -n C1 -- mkdir -p Partie_4
lxc_attach -n C1 -- touch Partie_4/operations.txt
echo Le fichier qui permet de sauvegarder les opérations réalisées a été créé.
```

- Le script `etat_synchro.sh` permet de garder sur les clients l'état des synchronisations dans le fichier `ETAT.txt`.

```
#! /bin/bash
if [ $# -ne 1 ]; then
    echo "Il manque le nom du conteneur où le fichier ETAT.txt doit être créé."
else
    date=$(date +'%M:%H:%d:%m')
    phrase="$date Fichier créé --> Aucune synchronisation "
    nom=$1
    lxc-attach -n $nom -- mkdir -p Partie_4
    lxc-attach -n "$nom" -- bash -c "echo '$phrase' >> Partie_4/ETAT.txt"
    echo Le fichier permettant de garder sur les clients l état des synchronisations  a été créé.
fi
```

- Le script `synchroniser.sh` remplace la version la plus ancienne du dossier ou du fichier avec la version la plus récente en appelant le script décrit au point suivant.

```
#! /bin/bash
# Fonction récursive pour traiter les sous-dossiers
recursive_comparison() {
    local dossier="$1"
    readarray -d '' resultats < <(find "$dossier" -type d -o -type f -print0)
    for element in "${resultats[@]}"; do
        nom_fichier=$(basename "$element")

        if [[ "$nom_fichier" == *.txt ]]; then
            if [[ "$element" =~ SYNCH ]]; then
                chemin=$(echo "$element" | sed 's/.*SYNCH\///')
            else
                chemin=$(echo "$element" | sed 's/.*Partie_4\///')
            fi

            ./comparaison.sh C1 C2 "$chemin" true
            ./comparaison.sh C1 C3 "$chemin" true
            ./comparaison.sh C3 C2 "$chemin" true
```

```
        else
            recursive_comparison "$element"
        fi
    done
}


if [ $# -ne 1 ]
then
    echo "Le nombre d argument n est pas bon --  [1 : nom du fichier ou du
dossier (avec son PATH après Partie_4/SYNCH en entier) ]"
    exit 1
else
    fichier_dossier=$1
    date=$(date +'%M:%H:%d:%m')
    sur_C2=false
    sur_C3=false

    #on vérifie si le dossier/fichier existe dans le fichier SYNCH.txt de C2
    while IFS= read -r line; do
    last_part=$(echo "$line" | awk '{print $NF}')
    if [ "$fichier_dossier" = "$last_part" ]; then
        #echo présent sur C2
        lxc-attach -n C2 -- bash -c "echo '$date synchronisation du fichier
$fichier_dossier présent sur le serveur. ' >> /Partie_4/ETAT.txt"
        sur_C2=true
        break
    fi
    done < /var/lib/lxc/C2/rootfs/Partie_4/SYNCH.txt

    #on vérifie si le dossier/fichier existe dans le fichier SYNCH.txt de C3
    while IFS= read -r line; do
    last_part=$(echo "$line" | awk '{print $NF}')
    if [ "$fichier_dossier" = "$last_part" ]; then
        #echo présent sur C3
        lxc-attach -n C3 -- bash -c "echo '$date synchronisation du fichier
$fichier_dossier présent sur le serveur. ' >> /Partie_4/ETAT.txt"
        sur_C3=true
        break
    fi
    done < /var/lib/lxc/C3/rootfs/Partie_4/SYNCH.txt

    if [ "$sur_C2" = true ] && [ "$sur_C3" = true ]; then
        #echo 0
        if [[ "$fichier_dossier" == *.txt ]]; then
            ./comparaison.sh C1 C2 "$fichier_dossier" false
            ./comparaison.sh C1 C3 "$fichier_dossier" false
            ./comparaison.sh C3 C2 "$fichier_dossier" false

        else
            #on le fait pour chaque conteneur au cas où des fichiers ont été
rajoutés
            recursive_comparison
"/var/lib/lxc/C1/rootfs/Partie_4/SYNCH/$fichier_dossier"
            recursive_comparison
"/var/lib/lxc/C2/rootfs/Partie_4/$fichier_dossier"
            recursive_comparison
"/var/lib/lxc/C3/rootfs/Partie_4/$fichier_dossier"
```

```
        fi

    elif [ "$sur_C2" = true ] && [ "$sur_C3" = false ]; then
        #echo 1
        if [[ "$fichier_dossier" == *.txt ]]; then
            ./comparaison.sh C1 C2 "$fichier_dossier" false
        else
            recursive_comparison
"/var/lib/lxc/C1/rootfs/Partie_4/SYNCH/$fichier_dossier"
            recursive_comparison
"/var/lib/lxc/C2/rootfs/Partie_4/$fichier_dossier"
        fi
    elif [ "$sur_C2" = false ] && [ "$sur_C3" = true ]; then
        #echo 2
        if [[ "$fichier_dossier" == *.txt ]]; then
           ./comparaison.sh C1 C3 "$fichier_dossier" false
        else
            recursive_comparison
"/var/lib/lxc/C1/rootfs/Partie_4/SYNCH/$fichier_dossier"
            recursive_comparison
"/var/lib/lxc/C3/rootfs/Partie_4/$fichier_dossier"
        fi
    elif [ "$sur_C2" = false ] && [ "$sur_C3" = false ]; then
        #echo 3
        lxc-attach -n C2 -- bash -c "echo '$date fichier/dossier $fichier_dossier
non présent en local. Done. ' >> /Partie_4/ETAT.txt"
        lxc-attach -n C3 -- bash -c "echo '$date fichier/dossier $fichier_dossier
non présent en local. Done. ' >> /Partie_4/ETAT.txt"

    fi

fi
```

- Ce dernier script appelle `comparaison.sh` qui compare les versions et copie la version la plus récente sur le conteneur ayant la version une version ultérieure.

```
#! /bin/bash
if [ $# -ne 4 ]
then
    echo "Le nombre d argument n est pas bon -- [1 : nom du conteneur 1
(potentiellement Serveur) ] [2 : nom du conteneur 2] [3 :  nom fichier/dossier]
[4 : appel récursif]"
    exit 1
else
    conteneur1=$1
    conteneur2=$2
    fichier_dossier=$3
    recur=$4
    date=$(date +'%M:%H:%d:%m')

    #on récupère les paths
    if [ "$conteneur1" = "C1" ]; then
        path1="/var/lib/lxc/C1/rootfs/Partie_4/SYNCH/"
    else
        path1="/var/lib/lxc/$conteneur1/rootfs/Partie_4/"
    fi
```

```
    path2="/var/lib/lxc/$conteneur2/rootfs/Partie_4/"

    #on vérifie que les fichiers existent
    imp=false
    if [ -e "$path1/$fichier_dossier" ] ; then
        if [ -e "$path2/$fichier_dossier" ] ; then
            echo ...
        else
            cp $path1/$fichier_dossier $path2/$fichier_dossier
        fi
    else
        if [ -e "$path2/$fichier_dossier" ] ; then
            cp $path2/$fichier_dossier $path1/$fichier_dossier
        else
            echo ..
            imp=true
        fi
    fi

    if ! $imp; then
        #on compare les fichiers
        if cmp -s "$path1/$fichier_dossier" "$path2/$fichier_dossier"; then
                #echo "Les fichiers sont identiques."

                #On pense a rajouter les logs dans le fichiers
                if [ "$conteneur1" = "C1" ]; then
                    lxc-attach -n C1 -- bash -c "echo '$date Le fichier
$fichier_dossier était déjà bien à jour avec $conteneur2. Update des dates faite.
' >> /Partie_4/operations.txt"
                else
                    lxc-attach -n "$conteneur1" -- bash -c "echo '$date ---------
---- $fichier_dossier déjà bien synchonisé avec $conteneur2. ' >>
/Partie_4/ETAT.txt"
                    lxc-attach -n "$conteneur1" -- bash -c "echo '$date ---------
---- update de la date fait. ' >> /Partie_4/ETAT.txt"
                    lxc-attach -n "$conteneur1" -- bash -c "echo '$date ---------
---- done. ' >> /Partie_4/ETAT.txt"
                fi
                lxc-attach -n "$conteneur2" -- bash -c "echo '$date -------------
$fichier_dossier déjà bien synchonisé avec $conteneur1. ' >> /Partie_4/ETAT.txt"
                lxc-attach -n "$conteneur2" -- bash -c "echo '$date -------------
update de la date fait. ' >> /Partie_4/ETAT.txt"
                lxc-attach -n "$conteneur2" -- bash -c "echo '$date -------------
done. ' >> /Partie_4/ETAT.txt"

        else

            #echo "Les fichiers sont différents, comparaison des dates."

            timestamp1=$(stat -c %Y "$path1/$fichier_dossier")
            timestamp2=$(stat -c %Y "$path2/$fichier_dossier")

            if [ "$timestamp1" -eq "$timestamp2" ]; then
                #echo "Les dates de création des fichiers sont identiques."

                #On pense a rajouter les logs dans le fichiers
                if [ "$conteneur1" = "C1" ]; then
```

```
                    lxc-attach -n C1 -- bash -c "echo '$date Le fichier
$fichier_dossier a été créé en même temps et est pourtant différent sur
$conteneur2. Problème. ' >> /Partie_4/operations.txt"
                else
                    lxc-attach -n "$conteneur1" -- bash -c "echo '$date ---------
---- Le fichier $fichier_dossier a été créé en même temps et est pourtant
différent sur $conteneur2. Problème. ' >> /Partie_4/ETAT.txt"
                    lxc-attach -n "$conteneur1" -- bash -c "echo '$date ---------
---- done. ' >> /Partie_4/ETAT.txt"
                fi
                lxc-attach -n "$conteneur2" -- bash -c "echo '$date -------------
Le fichier $fichier_dossier a été modifié en même temps et est pourtant différent
sur $conteneur1. Problème. ' >> /Partie_4/ETAT.txt"
                lxc-attach -n "$conteneur2" -- bash -c "echo '$date -------------
done. ' >> /Partie_4/ETAT.txt"

            elif [ "$timestamp1" -lt "$timestamp2" ]; then
                #echo "le fichier/document sur $conteneur1 a été créé avant."

                #On copie
                cp -r $path2/$fichier_dossier $path1/$fichier_dossier

                #On pense a rajouter les logs dans le fichiers
                if [ "$conteneur1" = "C1" ]; then
                    lxc-attach -n C1 -- bash -c "echo '$date Le fichier
$fichier_dossier est plus récent sur $conteneur2. Update du fichier local.' >>
/Partie_4/operations.txt"
                else
                    lxc-attach -n "$conteneur1" -- bash -c "echo '$date ---------
---- Le fichier $fichier_dossier est plus récent sur $conteneur2. Update du
fichier local. ' >> /Partie_4/ETAT.txt"
                    lxc-attach -n "$conteneur1" -- bash -c "echo '$date ---------
---- done. ' >> /Partie_4/ETAT.txt"
                fi
                lxc-attach -n "$conteneur2" -- bash -c "echo '$date -------------
Le fichier $fichier_dossier est plus récent en local. Update sur $conteneur1. '
>> /Partie_4/ETAT.txt"
                lxc-attach -n "$conteneur2" -- bash -c "echo '$date -------------
done. ' >> /Partie_4/ETAT.txt"


            else
                #echo "le fichier/document sur $conteneur2 a été créé avant."

                #On copie
                cp -r $path1/$fichier_dossier $path2/$fichier_dossier

                #On pense a rajouter les logs dans le fichiers
                if [ "$conteneur1" = "C1" ]; then
                    lxc-attach -n C1 -- bash -c "echo '$date Le fichier
$fichier_dossier est plus récent en local. Update sur $conteneur2.' >>
/Partie_4/operations.txt"
                else
                    lxc-attach -n "$conteneur1" -- bash -c "echo '$date ---------
---- Le fichier $fichier_dossier est plus récent en local. Update sur
$conteneur2. ' >> /Partie_4/ETAT.txt"
                    lxc-attach -n "$conteneur1" -- bash -c "echo '$date ---------
---- done. ' >> /Partie_4/ETAT.txt"
```

```
                fi
                lxc-attach -n "$conteneur2" -- bash -c "echo '$date -------------
Le fichier $fichier_dossier est plus récent sur $conteneur1. Update du fichier
local. ' >> /Partie_4/ETAT.txt"
                lxc-attach -n "$conteneur2" -- bash -c "echo '$date -------------
done. ' >> /Partie_4/ETAT.txt"


            fi
        fi
        #On met à jours les dates
        if [ "$recur" = "false" ]; then
            sed -i "/.* $fichier_dossier/d"
/var/lib/lxc/$conteneur1/rootfs/Partie_4/SYNCH.txt
            echo "$date $fichier_dossier" >>
/var/lib/lxc/$conteneur1/rootfs/Partie_4/SYNCH.txt

            sed -i "/.* $fichier_dossier/d"
/var/lib/lxc/$conteneur2/rootfs/Partie_4/SYNCH.txt
            echo "$date $fichier_dossier" >>
/var/lib/lxc/$conteneur2/rootfs/Partie_4/SYNCH.txt
        fi
    fi
fi
```

c. Créer un script depuis la machine hôte pour piloter l'exemple d'exécution.

Le script principal s'appelle MAIN.sh.

```
#! /bin/bash
# création des dossiers et fichiers nécessaires
./crea_fichier_synch.sh C1
./crea_fichier_synch.sh C2
./crea_fichier_synch.sh C3
./fichier_serveur.sh
./etat_synchro.sh C2
./etat_synchro.sh C3

#Appelle du script GENERE.sh qui permet de créer des dossiers et des fichiers
pour les tests
./GENERE.sh

#on ajoute les fichiers/dossier à synchroniser
    #C2 ajoute un fichier a synchoniser
    ./ajout_dossier_fichier.sh C2 C2_history.txt

    #C3 ajoute un fichier différent au précédent que C2 n'a pas à synchoniser
    ./ajout_dossier_fichier.sh C3 C3_history.txt

    #C2 et C3 ajoute un dossier ayant le même nom à synchroniser qui sont
identiques
    ./ajout_dossier_fichier.sh C2 Our_History
    ./ajout_dossier_fichier.sh C3 Our_History

#on appelle le script de synchronisation --> déjà identique au serveur
./synchroniser.sh C2_history.txt
sleep 3
```

```
#on appelle le script de synchronisation --> on modifie la version du client
echo 'Mon histoire en tant que SUPER conteneur C3 est ... ' >>
/var/lib/lxc/C3/rootfs/Partie_4/C3_history.txt
./synchroniser.sh C3_history.txt
sleep 3
#on appelle le script de synchronisation --> on modifie la version du serveur
echo 'Mon histoire en tant que super conteneur C3 est ... ' >>
/var/lib/lxc/C1/rootfs/Partie_4/SYNCH/C3_history.txt
./synchroniser.sh C3_history.txt

#on appelle le script de synchronisation --> on vérifie récursivement que les
dossiers contiennent les mêmes choses
sleep 3
./synchroniser.sh Our_History
#to do rajouter update dans synch pour les dossiers

#on appelle le script de synchronisation --> on vérifie récursivement que les
dossiers contiennent les mêmes choses on modifie middle.txt et on rajoute end.txt
sur un autre
sleep 3
echo 'par où commencer... ' >>
/var/lib/lxc/C2/rootfs/Partie_4/Our_History/middle.txt
./synchroniser.sh Our_History

#on rajoute un fichier dans Our_history sur C3
sleep 3
echo 'et la fin n est pas prete d arrvier ' >>
/var/lib/lxc/C3/rootfs/Partie_4/Our_History/end.txt
./synchroniser.sh Our_History
```

Il appelle tous les autres scripts. Il appelle le script GENERE.sh qui créé des fichiers et des dossiers sur les clients/serveurs pour tester.

```
#! /bin/bash

#C2 a un fichier C2_history.txt
echo 'Mon histoire en tant que conteneur C2 est ... ' >>
/var/lib/lxc/C2/rootfs/Partie_4/C2_history.txt

#C3 a un fichier C3_history.txt
echo 'Mon histoire en tant que conteneur C3 est ... ' >>
/var/lib/lxc/C3/rootfs/Partie_4/C3_history.txt

#C2 et C3 ont le même dossier Our_History/start.txt et Our_History/middle.txt
mkdir -p /var/lib/lxc/C2/rootfs/Partie_4/Our_History
mkdir -p /var/lib/lxc/C3/rootfs/Partie_4/Our_History

echo 'Le début a été ... ' >>
/var/lib/lxc/C2/rootfs/Partie_4/Our_History/start.txt
echo 'Le début a été ... '
>>/var/lib/lxc/C3/rootfs/Partie_4/Our_History/start.txt

echo 'Le milieu a été ... '
>>/var/lib/lxc/C2/rootfs/Partie_4/Our_History/middle.txt
echo 'Le milieu a été ... '
>>/var/lib/lxc/C3/rootfs/Partie_4/Our_History/middle.txt
```

Le script `CLEAN.sh` m'a permis de nettoyer afin de recommencer mes tests :

```
#! /bin/bash
rm -r /var/lib/lxc/C1/rootfs/Partie_4
rm -r /var/lib/lxc/C2/rootfs/Partie_4
rm -r /var/lib/lxc/C3/rootfs/Partie_4
```

d. Une fois les scripts écrits effectuer un exemple d'exécution

On remarque que le serveur contient bien dans son dossier `SYNCH` tous les fichiers que l'on souhaitait retrouver.

```
root@srv:/var/lib/lxc/C1/rootfs# cd Partie_4/
root@srv:/var/lib/lxc/C1/rootfs/Partie_4# ls
SYNCH  SYNCH.txt  operations.txt
root@srv:/var/lib/lxc/C1/rootfs/Partie_4# cd SYNCH
root@srv:/var/lib/lxc/C1/rootfs/Partie_4/SYNCH# ls
C2_history.txt  C3_history.txt  Our_History
root@srv:/var/lib/lxc/C1/rootfs/Partie_4/SYNCH# ls Our_History/
end.txt  middle.txt  start.txt
root@srv:/var/lib/lxc/C1/rootfs/Partie_4/SYNCH#
```

Le fichier `operation.txt` comporte bien toutes les opérations réalisées :

```
root@srv:/var/lib/lxc/C1/rootfs/Partie_4# cat operations.txt
07:12:11:06 Le fichier C2_history.txt était déjà bien à jour avec C2. Update des dates faite.
07:12:11:06 Le fichier C3_history.txt est plus récent sur C3. Update du fichier local.
07:12:11:06 Le fichier C3_history.txt est plus récent en local. Update sur C3.
07:12:11:06 Le fichier Our_History/start.txt était déjà bien à jour avec C2. Update des dates faite.
07:12:11:06 Le fichier Our_History/start.txt était déjà bien à jour avec C3. Update des dates faite.
07:12:11:06 Le fichier Our_History/middle.txt était déjà bien à jour avec C2. Update des dates faite.
```

Les fichiers de synchronisations `SYNCH.txt` fonctionnent correctement :

```
07:12:11:06 Our_History
07:12:11:06 C2_history.txt
07:12:11:06 C3_history.txt
```

Enfin on retrouve bien sur les clients le fichier ETAT.txt comportement seulement les informations qui sont liées au conteneur qui détient ce fichier :

```
root@srv:/var/lib/lxc/C2/rootfs/Partie_4# cat ETAT.txt
07:12:11:06 Fichier créé --> Aucune synchronisation
07:12:11:06 synchronisation du fichier C2_history.txt présent sur le serveur.
07:12:11:06 ------------- C2_history.txt déjà bien synchonisé avec C1.
07:12:11:06 ------------- update de la date fait.
07:12:11:06 ------------- done.
```