sync scripts in ~/bin via Nextcloud between Linux systems

I use Nextcloud to store and synchronize my files between my devices. These also include a couple of Linux PC's. I'm not using a roaming profile, so my home directory is local to every machine and that is okay, because they are different machines with different setups. However, over the time i've used linux, i have collected some handy bash scripts for my daily work which facilitate a couple of repeditive tasks and i would like to have those tools synced across all my linux machines, so they are updated on all of them if I add a new one or enhance an existing one.

I have therefore created a folder named `homebin` in my Nextcloud account, which is then synced using the Nextcloud client to `~/Nextcloud/homebin`.

The Problem

The problem with Nextcloud is, that it does not sync permissions of files, so all the files in the `~/Nextcloud/hoembin` could be made executable locally, but when synced, would not be executable on the other machines. Initially, i created an alias for each script in my `.bash_aliases` file but that became cumbersome as I have to add new entries for each script on all of my PC's.

My Solution

I recently found a tool called posixovl which stands probably for POSIX Overlay :)

this tool allows us to turn a non-posix capable directory into a posix capable mountpoint. Its main purpose is to add POSIX features to VFAT formatted filesystems, however, it also works on top of filesystems that are already posix capable (like in our case).

on ubuntu, posixovl is available via the repos, so it can simply be installed using

```
sudo apt install fuse-posixovl
```

unfortunately though, posixovl does not provide a mount compatible `/sbin/mount.posixovl`. Even worse, the ubuntu package at least provides this file, but it is the `mount.posixovl` which does not accept the kind of parametners that mount would call. So out of the box it is not possible to use posixovl via fstab which is cumbersome.

Since I wanted this directory to be mounted automatically, I frist tried to write a startup script that would mount the directory upon login to gnome. This works for logging in, however it fails miserably after a logout. The directory stays mounted and becomes inaccessible to the user. I tried to add an unmount in `/etc/gdm/PostScripts/Default` but that didn't work either.

I found the solution on stackexchange which almost worked.. there where to issues with the solution (at least that's what I thaught):

1. the original binary had to be renamed –> cumbersome when doing updates in the future through apt
2. mount.posixovl.orig was started as root, making the mount inaccessible to my user. passing the `uid` argument for fuse did not seem to make a difference

so i slightly modified the script and saved it to `/sbin/mount.fuse-posixovl` so that i can use a filesystem type `fuse-posixovl` in my fstab and did not need to rename the original file so updates will still work.

mount.fuse-posixovl

```
#!/bin/bash
# wrapper for mount.posixovl to conform with common mount syntax
# with this wrapper posixovl can be used in fstab

# location of the original mount.posixovl
origposixovl="/sbin/mount.posixovl"

# gather inputs
while [ $# -gt 0 ]; do
        if [[ "$1" == -* ]]; then
                # var is an input switch
                # we can only use the -o or -F switches
                if [[ "$1" == *F* ]]; then
                        optsF="-F"
                else
                        optsF=""
                fi
                if [[ "$1" == *o* ]]; then
                        shift
                        if [[ "$1" == *uid=* ]]; then
                                runas=$(getent passwd $(echo "$1" | sed
-E -e 's/^.*uid=([^,]+)(,.*)?$/\1/') | cut -d: -f1)
                        fi
                        optsfuse="-- -o $1"
                else
                        optsfuse=""
                fi
                shift
        else
                # var is a main argument
                sourcedir="$1"
                shift
                if [[ "$1" != -* ]]; then
                        targetdir="$1"
                        shift
                else
                        targetdir="$sourcedir"
                fi
        fi
```

```
done

# verify inputs
if [ "$sourcedir" == "" ]; then
        echo "no source specified"
        exit 1
fi
if [ "$targetdir" == "" ]; then
        echo "no target specified"
        exit 1
fi

# build mount.posixovl command
if [[ -n "$runas" ]]; then
        su - "${runas}" -c "\"$origposixovl\" $optsF -S \"$sourcedir\"
\"$targetdir\" $optsfuse"
else
        "$origposixovl" $optsF -S "$sourcedir" "$targetdir" $optsfuse
fi
```

now I added this line to fstab and voilà :)

```
/home/myuser/Nextcloud/homebin            /home/myuser/bin fuse-posixovl
uid=1000,gid=1000    0 0
```

(i don't think the `gid=1000` is doing anything.. but i put it in there anyway)

"why didn't you..."

... simply create a startup script that runs `chmod +x ~/Nextcloud/homebin/*` on startup? Well, that would have worked and been easier, but i am not very good on shutting down my systems, so some of them run for months and laptops go to standby between active sessions, so the login scripts aren't executed all that often. If i add new scripts in the meantime, they will be synced via nextcloud but won't be executable then.

... create a cron job that runs the above `chmod` every few minutes? That would have been possible and it probably would have solved the problem sufficiently for this specific case, but i have some other directories where I also want to be able to store posix properties, and there not all the files should be executable. The posixovl solution allows to “disalbe” scripts to make sure they are not accidently executed (they won't appear in tab completion either) and it works instantaneously, as soon as the script is synced it is also executable on all machines. so all in all just more flexible and elegant :)