Commander la LED avec un bouton poussoir

Cahier des charges : Lorsqu'on appuie sur le bouton poussoir, la LED s'allume. Elle reste allumée tant qu'on appuie sur le bouton poussoir. Elle s'éteint quand on relâche le bouton poussoir.

Ouvrez l'éditeur de texte `nano` :

```
pi@raspberrypi ~ $ nano cde_LED_01.sh
```

Saisissez le script destiné à lire l'état d'un bouton poussoir connecté sur le GPIO16 et à commander une LED connectée sur le GPIO 20.

```
# !/bin/bash
# Script cde_LED_01.sh
# Lire l'état d'un bouton poussoir
# Et commander une LED
# Appuyé => LED allumée    Relaché => LED éteinte
# Ce script ne comporte pas de test destiné à l'interrompre
# Il s'arrête en appuyant sur CTRL  C

# Effacer l'écran
clear
echo "Appuyer sur le bouton pour allumer la LED"

# Rendre le répertoire /sys/class/gpio actif
cd /sys/class/gpio

############### Configuration port GPIO du bouton poussoir

# Créer l'accès au port GPIO 16
# Pour lire l'état du bouton poussoir
# Ne rien faire s'il existe déja
if [ -d "gpio16" ]; then
 echo "gpio16 existe dejà"
else
 echo "gpio16 : Création"
 echo 16 > export
fi

# Rendre le répertoire gpio16 actif
cd gpio16/

# Configurer le port GPIO 16 en entrée
# Normalement il y est par défaut… mais on ne sait pas ce qui a pu se passer avant
echo in > direction

############### Configuration port GPIO de la LED

# revenir dans le dossier /sys/class/gpio
cd ..

# Créer l'accès au port GPIO 20
# Pour commander la LED
# Ne rien faire s'il existe déja
if [ -d "gpio20" ]; then
 echo "gpio20 existe dejà"
else
 echo "gpio20 : Création"
```

```
 echo 20 > export
fi

# Rendre le répertoire gpio20 actif
cd gpio20/

# Configurer le port GPIO 20 en sortie
echo out > direction

# Eteindre la LED
echo 0 > value

############################  Boucle de scrutation du port GPIO 16

# Initialiser la variable x à 1
x="1"

    # L'instruction cat affiche le contenu d'un fichier à l'écran
    # Le fichier value contient l'état du bouton poussoir
    # Au lancement du programme on allume (ou pas) la LED selon l'état du bouton poussoir
        z="$(cat /sys/class/gpio/gpio16/value)"
#        echo "Bouton => " $z
        if [ $z = 1 ]; then
                echo 1 > /sys/class/gpio/gpio20/value
#                echo "Allumer la LED"
         else
                echo 0 > /sys/class/gpio/gpio20/value
#                echo "Eteindre la LED"
        fi


# Boucle while : cette boucle s'exécute tant que
# la condition est vérifiée
# ici la condition est toujours vraie :
# la boucle ne s'interrompt jamais
# Il faudra en sortir avec un CTRL C


while [ $x -gt 0 ]
# Le bloc inclus entre do et done est exécuté
# par la boucle while

# Début du bloc d'instructions
do
#echo "entrée dans la boucle"

    # On mémorise l'état du bouton poussoir dans la variable m
        let "m = z"

    # On lit à nouveau l'état du bouton poussoir
    # -ne  teste si non égal
        z="$(cat /sys/class/gpio/gpio16/value)"
#        echo "Dans la boucle bouton poussoir => " $z
        if [ "$z" -ne "$m" ]; then
       # Si l'état est différent du précédent, on affiche l'état actuel
            if [ $z = 1 ]; then
                echo 1 > /sys/class/gpio/gpio20/value
#                echo "Etat différent => Allumer"
         else
                echo 0 > /sys/class/gpio/gpio20/value
```

```
#                echo "Etat différent => Eteindre"
            fi
      # Fin du premier if
        fi

    # La commande sleep suspend l'exécution du programme
    # pendant la durée spécifiée (en secondes)
    sleep 0.1

# Fin du bloc d'instructions
done
```