waitlock

```
while test –r lockfile ; do sleep 5 ; done
#                       ^ caractère «new line»
```

Ce programme attend qu'un fichier soit détruit.

Que fait ce programme ?

```
until test –r stopfile
    do sleep 2
    echo Hello
done &          # & pour exécution en arrière-plan
```

Créer un fichier nommé stopfile ; que se passe-t-il ?

---

fruits

```
for fruits in apples pears oranges mangos
do                              #do exécuté pour chaque élément dans liste
 echo $fruits are fruits
done
```

---

breeds

```
for breed                       # exécute le do pour chaque paramètre
do
 case $breed in             # ;; fin des commandes exécutées pour ce cas
      arabian|palomino|clydesdale) echo $breed is a horse ;;
      jersey|guernsey|holstein) echo $breed is a cow ;;
      husky|shepherd|setter|labrador) echo $breed is a dog ;;
      siamese|persian|angora|) echo $breed is a cat ;;
      *) echo $breed is not in our catalog ;;
 esac
done
```

Exécuter :

```
    breeds husky holstein terrier
```

---

lsdir6

```
# liste les sous-répertoires des répertoires donnés en paramètres
# usage : lsdir6 [rép1/ rép2/ ... répn/]
#
if test $# = 0          #    $# variable contenant le nombre de
paramètres
    then
       param=.
    else                #    $@ variable contenant les paramètres sous
       param="$@"       #    forme d'une liste, équivalent $1 $2 ... $n
fi
for i in $param
do
    for j in $i/*       #    $i/* la liste des éléments dans le ne
répertoire
    do
        if test -d $j
            then
                echo $j
        fi
    done
done
```

fna

```
# utilisation de fonctions
TMP=/tmp
PIDLOG=${TMP}/pidlog$$
DATEFILE==${TMP}/date$$
ERRLOG=${TMP}/demolog
#
# fonction shell pour nettoyer et quitter
errexit() {
 echo $1
 date >> $ERRLOG
 echo $1 >> $ERRLOG
 rm -f $PIDLOG $DATEFILE
 exit
}
#
# fonction shell pour lire une réponse y/n et fixer code retour 0 ou 1

ok() {
 while true
 do
       read ans
       case $ans in
            [yY]*)   return 0 ;;
            [nN]*)   return 1 ;;
            *)   echo Please answer y or n ;;
       esac
 done
}
# début traitement de la commande
echo $$ > $PIDLOG          # $$ variable contenant le PID du shell
date > $DATEFILE
#
# test error exit
echo Test errexit function [y/n]?
ok && errexit "Testing the errexit function"
#  ^^ pipeline exécuté si vrai (|| utilsé si faux)
echo Normal Termination
echo Please remove $PIDLOG and $DATEFILE
```

Tester en exécutant fna et en répondant successivement ***y***, ***maybe*** et ***n***.