Fonction arroser

Version impérative

```
#! /bin/bash
arroser () {
    echo "Je m'amuse tellement à faire du bash"
}
arroser
```

```
#! /bin/bash
arroser () {
    while read line; do
        echo $line
    done <plante
}
arroser
```

```
#! /bin/bash
if [ $# -ne 1 ]; then
  echo "Please provide exactly one argument" >&2 # &2 est la sortie   d erreur
  exit 1 # un code de retour faux
fi
arroser () {
    echo $1 >plante
}
arroser $1
```

```
#! /bin/bash
if [ $# -ne 1 ]; then
  echo "Please provide exactly one argument" >&2 # &2 est la sortie   d erreur
  exit 1 # un code de retour faux
fi
boucleImp () {
    for i in `seq 1 $1`; do
        read k <plante
        k=$(expr $k + 1)
        echo $k > plante
    done
}
arroser () {
    boucleImp $1
}
arroser $1
```

Version récursive

```
    boucleRec () {
        if [ $1 -ne 0 ]; then
            k=$1
            k=$(expr $k - 1)
            echo $k
```

```
6               boucleRec $k
7           else
8               echo "X n'est plus"
9           fi
10      }
```

```
1  #! /bin/bash
2
3  if [ $# -ne 1 ]; then
4    echo "Please provide exactly one argument" >&2 # &2 est la sortie   d erreur
5    exit 1 # un code de retour faux
6  fi
7
8  boucleImp () {
9      for i in `seq 1 $1`; do
10         read k <plante
11         k=$(expr $k + 1)
12         echo $k > plante
13     done
14 }
15
16 boucleRec () {
17     if [ $1 -ne 0 ]; then
18         a=$1
19         a=$(expr $a - 1)
20         echo $a
21
22         read k <plante
23         k=$(expr $k + 1)
24         echo $k > plante
25
26         boucleRec $a
27     else
28         echo "X n'est plus"
29     fi
30 }
31
32 #boucleImp $1
33 boucleRec $1
```

Auprès de mon arbre

```
1  #! /bin/bash
2
3  if [ "$#" -ne 1 ]; then
4      echo "Un seul paramètre requis" >&2
5  elif [ -d $1 ]; then
6      echo "Dossier : $1"
7  elif [ -e $1 ]; then
8      echo "Fichier : $1"
9  else
10     echo "Argument non valable"
11 fi
```

```
#! /bin/bash

if [ "$#" -ne 1 ]; then
    echo "Un seul paramètre requis" >&2
else
    echo "Argument non valable" >&2
fi

parseRec () {
    cd $1

    liste_fichiers=`ls`

    for file in $liste_fichiers; do
        if [ -d $file ]; then
            echo "---- $file" >&1
            parseRec $file
        elif [ -e $file ]; then
            echo "---- $file" >&1
        fi
    done
    cd ..
}

parseRec $1
```

```
#! /bin/bash

if [ "$#" -ne 1 ]; then
    echo "Un seul paramètre requis" >&2
fi

parseRec () {
    cd $1
    cpt=$(expr $cpt + 1)

    liste_fichiers=`ls`

    for file in $liste_fichiers; do
        for i in `seq 2 $cpt`; do
            printf " ---- "
        done

        if [ -d $file ]; then
            echo "---- $file" >&1
            parseRec $file $cpt
        elif [ -e $file ]; then
            echo "---- $file" >&1
        fi
    done

    cd ..
    cpt=$(expr $cpt - 1)
}

cpt=0
parseRec $1 $cpt
```

Mise en évidence desincohérences provoquées par les commutations

écriture.sh :

```
#! /bin/bash

if [ $# -lt 1 ] ; then
    echo "Il faut au moins un parametre"
    exit 1
fi
for elem in "$@" ; do
    if [ ! -e "$elem" ] ; then
        echo premier $$ > "$elem"
    else
        echo suivant $$ >> "$elem"
    fi
done
```

lancement_écriture.sh :

```
#! /bin/bash

rm -f f1 f2 f3

./ecriture.sh f1 f2 f3 & ./ecriture.sh f1 f2 f3

wait

cat f1
cat f2
cat f3
```