

PROGRAMMATION UNIX

Notes de Cours/TD/TP autorisées; autres documents, calculatrices, ordinateurs interdits.

Ne répondez pas à plusieurs questions en même temps; respectez le découpage de l'énoncé, sous peine de nullité. Certaines questions sont plus faciles que d'autres, lisez bien l'énoncé avant de commencer.

I. Script bash gérant un catalogue de jeux (20 points)

On se propose d'écrire le script bash `cataga.sh` pour gérer un catalogue de jeux installés sur une machine. Pour chaque jeu répertorié dans ce catalogue il y a un petit script bash, appelé un *lanceur*, qui sait afficher des informations sur le jeu et qui sait l'exécuter.

1) Les lanceurs sont tous situés dans un même répertoire, mémorisé dans la variable globale `CATAREP`. On suppose que cette variable est initialisée au début du script.

Écrire la fonction `CreerRepLanceurs`, ne prenant pas d'argument. La fonction vérifie l'existence du répertoire des lanceurs, sinon elle crée le répertoire et affiche un message de création; si cette création échoue, la fonction affiche un message d'erreur et fait échouer le script.

2) Les lanceurs sont tous de la forme `"ga-x.sh"` où x est un entier positif ou nul, appelé le *numéro* du lanceur.

Écrire la fonction `ConstruireCheminLanceur`, prenant en argument un numéro de lanceur. Sans vérifier si ce numéro est un entier, la fonction affiche sur la sortie standard le chemin complet du lanceur.

Par exemple si `CATAREP` est `"$HOME/.catarep/"`, votre home-directory est `/home/toto` et le numéro x est 23, la fonction affichera sur la sortie standard `"/home/toto/.catarep/ga-23.sh"`.

3) Écrire la fonction `EstEntier`, prenant en argument un mot x . La fonction réussit si x est un entier positif ou nul, échoue sinon.

4) Écrire la fonction `AssurerEntier`, prenant en argument un mot x . Si x est un entier positif ou nul, la fonction affiche x sur la sortie standard, sinon elle affiche "0".

5) Écrire la fonction `ExtraireNumeroLanceur`, prenant en argument un chemin complet de lanceur. La fonction en extrait le numéro de lanceur et l'affiche sur la sortie standard. Si le numéro extrait n'est pas *valide*, c'est-à-dire n'est pas un entier positif, la fonction affiche "0".

6) Écrire la fonction `MaxNumeroLanceur`, ne prenant pas d'argument. La fonction affiche sur la sortie standard le plus grand numéro des lanceurs présents dans le répertoire des lanceurs. Si ce répertoire est vide, alors la valeur affichée sera "-1".

7) Écrire la fonction `IncrementerLanceurs`, prenant en argument un numéro de lanceur x (si x est invalide on considère que le numéro est 0). La fonction renomme tous les lanceurs dont le numéro est supérieur ou égal à x , en incrémentant leur numéro de 1.

8) Écrire la fonction `CreerLanceur`, prenant en argument un numéro de lanceur x (si x est invalide on considère que le numéro est 0).

La fonction demande interactivement à l'utilisateur de rentrer successivement le nom du jeu, une légende (c'est-à-dire une courte description en une ligne), puis la commande à exécuter pour lancer le jeu. La fonction génère ensuite le lanceur numéro x . Voici le code généré sur un exemple,

```
#!/bin/bash
#
# Lanceur généré par ./cataga.sh
# le lundi 19 décembre 2011, 10:36:07 (UTC+0100)

case "${1:-}" in
  -nom)      echo "tangram" ;;
  -legende)  echo "Jeu de Tangram" ;;
  -lancer)   exec $HOME/ez-draw/jeu-tangram ;;
  *)        echo "$0: mauvais paramètre" 1>&2 ;;
esac

exit 0
```

dans lequel le jeu s'appelle "tangram", la légende est "Jeu de Tangram", la commande pour lancer le jeu est "\$HOME/ez-draw/jeu-tangram"; à vous de substituer correctement ces valeurs ainsi que le nom du script et la date de génération. Enfin, la fonction rend le lanceur exécutable.

9) Écrire la fonction `LancerJeu`, prenant en argument un numéro de lanceur x (si x est invalide on considère que le numéro est 0). La fonction vérifie que le lanceur numéro x existe et est exécutable, puis l'exécute en tâche de fond avec pour mission de lancer le jeu correspondant.

10) Écrire la fonction `ConstruireMenu`, ne prenant pas d'argument. La fonction construit dans le répertoire des lanceurs un fichier de menu, dont le chemin complet est supposé mémorisé dans la variable globale `CATAMENU`. Ce fichier est un fichier texte dont chaque ligne correspond à un lanceur. Le format d'une ligne est : le numéro de lanceur, puis " : ", puis le nom du jeu, puis " - ", puis la légende. Le fichier est trié sur l'ordre croissant des numéros de lanceurs. Voici un exemple :

```
4 : bubblet - Jeu de Bulles
7 : tangram - Jeu de Tangram
12 : taquin - Puzzle coulissant
```

11) On suppose disposer (ne pas les écrire!) des fonctions `AfficherMenu` et `AfficherUsage`. La première affiche le contenu du menu des lanceurs sur la sortie standard (sinon un message d'erreur); la seconde affiche l'usage du script sur la sortie standard :

```
USAGE: cataga.sh options ...

Options :
  --help      : affiche cette aide
  --menu      : affiche le menu des jeux
  --majm      : met à jour le menu des jeux
  --nouv n    : crée le lanceur numéro n
  --incr n    : incrémente les numéros de lanceurs >= n
  --joue n    : lance le jeu numéro n
```

Écrire le programme principal du script, qui vérifie la présence d'arguments, sinon il affiche l'usage sur la sortie d'erreur puis échoue. Le programme vérifie ensuite la présence du répertoire des lanceurs, au besoin il essaie de le créer. Le programme analyse ensuite les arguments et procède aux actions nécessaires (voir l'usage). Voici un exemple d'utilisation :

```
./cataga.sh --nouv 2 --incr 7 --nouv 7 --majm --menu --joue 2 --joue 4
```

Correction

I. Script bash gérant un catalogue de jeux (20 points)

On se propose d'écrire le script bash `cataga.sh` pour gérer un catalogue de jeux installés sur une machine. Pour chaque jeu répertorié dans ce catalogue il y a un petit script bash, appelé un *lanceur*, qui sait afficher des informations sur le jeu et qui sait l'exécuter.

1) Les lanceurs sont tous situés dans un même répertoire, mémorisé dans la variable globale `CATAREP`. On suppose que cette variable est initialisée au début du script.

Écrire la fonction `CreerRepLanceurs`, ne prenant pas d'argument. La fonction vérifie l'existence du répertoire des lanceurs, sinon elle crée le répertoire et affiche un message de création ; si cette création échoue, la fonction affiche un message d'erreur et fait échouer le script.

```
# CreerRepLanceurs
#
CreerRepLanceurs ()
{
    if [ ! -d "$CATAREP" ]; then
        echo "Création $CATAREP ..."
        if ! mkdir "$CATAREP" ; then
            echo "Echec création $CATAREP" 1>&2
            exit 1
        fi
    fi
}
```

2) Les lanceurs sont tous de la forme `"ga-x.sh"` où *x* est un entier positif ou nul, appelé le *numéro* du lanceur.

Écrire la fonction `ConstruireCheminLanceur`, prenant en argument un numéro de lanceur. Sans vérifier si ce numéro est un entier, la fonction affiche sur la sortie standard le chemin complet du lanceur.

Par exemple si `CATAREP` est `"$HOME/.catarep/"`, votre home-directory est `/home/toto` et le numéro *x* est 23, la fonction affichera sur la sortie standard `"/home/toto/.catarep/ga-23.sh"`.

```
# ConstruireCheminLanceur numéro
#
ConstruireCheminLanceur ()
{
    local numero="$1"
    echo "$CATAREP/ga-$numero.sh"
}
```

3) Écrire la fonction `EstEntier`, prenant en argument un mot *x*. La fonction réussit si *x* est un entier positif ou nul, échoue sinon.

```
# EstEntier mot
#
EstEntier ()
{
    local mot="$1"
    case "$mot" in
        *[^0-9]*) return 1 ;; # Échec
        *) return 0 ;; # Succès
    esac
}
```

4) Écrire la fonction `AssurerEntier`, prenant en argument un mot x . Si x est un entier positif ou nul, la fonction affiche x sur la sortie standard, sinon elle affiche "0".

```
# AssurerEntier mot
#
AssurerEntier ()
{
    local mot="$1"
    if ! EstEntier "$mot" ; then echo "0"
    else echo "$mot"
    fi
}
```

5) Écrire la fonction `ExtraireNumeroLanceur`, prenant en argument un chemin complet de lanceur. La fonction en extrait le numéro de lanceur et l'affiche sur la sortie standard. Si le numéro extrait n'est pas *valide*, c'est-à-dire n'est pas un entier positif, la fonction affiche "0".

```
# ExtraireNumeroLanceur chemin
#
ExtraireNumeroLanceur ()
{
    local chemin="$1"
    local suffixe="{chemin###/ga-}"
    local numero="{suffixe%.sh}"

    AssurerEntier "$numero"
}
```

6) Écrire la fonction `MaxNumeroLanceur`, ne prenant pas d'argument. La fonction affiche sur la sortie standard le plus grand numéro des lanceurs présents dans le répertoire des lanceurs. Si ce répertoire est vide, alors la valeur affichée sera "-1".

```
# MaxNumeroLanceur
#
MaxNumeroLanceur ()
{
    local max="-1" chemin numero

    for chemin in "$CATAREP"/ga-*.sh ; do
        numero=$(ExtraireNumeroLanceur "$chemin")
        if ((numero > max)); then max="$numero" ; fi
    done
    echo "$max"
}
```

7) Écrire la fonction `IncrementerLanceurs`, prenant en argument un numéro de lanceur x (si x est invalide on considère que le numéro est 0). La fonction renomme tous les lanceurs dont le numéro est supérieur ou égal à x , en incrémentant leur numéro de 1.

```
# IncrementerLanceurs numéro
#
# Il faut procéder dans l'ordre décroissant
# Solution 1 : à partir du max (coûteux si max est grand)
#
IncrementerLanceurs ()
{
    local numero=$(AssurerEntier "$1") max i ancien_nom nouveau_nom

    max=$(MaxNumeroLanceur)
```

```

    for (( i = max; i >= numero; i-- )); do
        ancien_nom=$(ConstruireCheminLanceur $i)
        nouveau_nom=$(ConstruireCheminLanceur $((i+1)))
        if [ -f "$ancien_nom" ]; then
            mv -fv "$ancien_nom" "$nouveau_nom"
        fi
    done
}

# IncrementerLanceurs numéro
# Solution 2 : à partir d'une liste triée dans l'ordre décroissant
#
IncrementerLanceurs2 ()
{
    local numero=$(AssurerEntier "$1") liste=()
    local chemin i ancien_nom nouveau_nom

    for chemin in "$CATAREP"/ga-*.sh ; do
        i=$(ExtraireNumeroLanceur "$chemin")
        if ((i >= numero)); then liste=(${liste[*]} $i) ; fi
    done

    echo ${liste[*]} | tr ' ' '\n' | sort -nr |
    while read i ; do
        ancien_nom=$(ConstruireCheminLanceur $i)
        nouveau_nom=$(ConstruireCheminLanceur $((i+1)))
        mv -fv "$ancien_nom" "$nouveau_nom"
    done
}

# IncrementerLanceurs numéro
# Solution 3 : en 2 passes avec renommage
#
IncrementerLanceurs3 ()
{
    local numero=$(AssurerEntier "$1")
    local chemin i nouveau_nom

    for chemin in "$CATAREP"/ga-*.sh ; do
        i=$(ExtraireNumeroLanceur "$chemin")
        if ((i < numero)); then continue ; fi
        nouveau_nom=$(ConstruireCheminLanceur $((i+1))).tmp"
        mv -fv "$chemin" "$nouveau_nom"
    done

    for chemin in "$CATAREP"/ga-*.sh.tmp ; do
        mv -fv "$chemin" "${chemin%.tmp}"
    done
}

```

8) Écrire la fonction `CreerLanceur`, prenant en argument un numéro de lanceur x (si x est invalide on considère que le numéro est 0).

La fonction demande interactivement à l'utilisateur de rentrer successivement le nom du jeu, une légende (c'est-à-dire une courte description en une ligne), puis la commande à exécuter pour lancer le jeu. La fonction génère ensuite le lanceur numéro x . Voici le code généré sur un exemple,

```

#! /bin/bash
#
# Lanceur généré par ./cataga.sh
# le lundi 19 décembre 2011, 10:36:07 (UTC+0100)

case "${1:-}" in
  -nom)      echo "tangram" ;;
  -legende)  echo "Jeu de Tangram" ;;
  -lancer)   exec $HOME/ez-draw/jeu-tangram ;;
  *)         echo "$0: mauvais paramètre" 1>&2 ;;
esac

exit 0

```

dans lequel le jeu s'appelle "tangram", la légende est "Jeu de Tangram", la commande pour lancer le jeu est "\$HOME/ez-draw/jeu-tangram"; à vous de substituer correctement ces valeurs ainsi que le nom du script et la date de génération. Enfin, la fonction rend le lanceur exécutable.

```

# CreerLanceur numéro
#
CreerLanceur ()
{
    local numero=$(AssurerEntier "$1") lanceur

    echo "Création lanceur numéro $numero :"
    echo -n "  Nom : " ; read nom
    echo -n "  Légende : " ; read legende
    echo -n "  Commande pour exécuter : " ; read commande

    lanceur=$(ConstruireCheminLanceur "$numero")
    cat >| "$lanceur" << FIN
#! /bin/bash
#
# Lanceur généré par $0 le $(date)

case "\${1:-}" in
  -nom)      echo "$nom" ;;
  -legende)  echo "$legende" ;;
  -lancer)   exec $commande ;;
  *)         echo "\$0: mauvais paramètre" 1>&2 ;;
esac

exit 0
FIN

    chmod +x "$lanceur"
}

```

9) Écrire la fonction `LancerJeu`, prenant en argument un numéro de lanceur x (si x est invalide on considère que le numéro est 0). La fonction vérifie que le lanceur numéro x existe et est exécutable, puis l'exécute en tâche de fond avec pour mission de lancer le jeu correspondant.

```

# LancerJeu numéro
#
LancerJeu ()
{
    local numero=$(AssurerEntier "$1") lanceur

    lanceur=$(ConstruireCheminLanceur "$numero")
    if [ -x "$lanceur" ]; then
        echo "Exécution du jeu numéro $numero ..."
        "$lanceur" -lancer &
    else
        echo "Erreur, jeu numéro $numero absent" 1>&2
    fi
}

```

10) Écrire la fonction `ConstruireMenu`, ne prenant pas d'argument. La fonction construit dans le répertoire des lanceurs un fichier de menu, dont le chemin complet est supposé mémorisé dans la variable globale `CATAMENU`. Ce fichier est un fichier texte dont chaque ligne correspond à un lanceur. Le format d'une ligne est : le numéro de lanceur, puis " : ", puis le nom du jeu, puis " - ", puis la légende. Le fichier est trié sur l'ordre croissant des numéros de lanceurs.

Voici un exemple :

```
4 : bubblet - Jeu de Bulles
7 : tangram - Jeu de Tangram
12 : taquin - Puzzle coulissant
```

```
# ConstruireMenu
# numéros triés par ordre croissant
#
ConstruireMenu ()
{
    local lanceur numero

    echo "Création de $CATAMENU ..."
    for lanceur in "$CATAREP"/ga*.sh ; do
        numero=$(ExtraireNumeroLanceur "$lanceur")
        if ((numero < 0)); then continue ; fi
        echo "$numero : $($lanceur -nom) - $($lanceur -legende)"
    done | sort -n >| "$CATAMENU"
}
```

11) On suppose disposer (ne pas les écrire!) des fonctions `AfficherMenu` et `AfficherUsage`. La première affiche le contenu du menu des lanceurs sur la sortie standard (sinon un message d'erreur); la seconde affiche l'usage du script sur la sortie standard :

```
USAGE: cataga.sh options ...

Options :
--help    : affiche cette aide
--menu    : affiche le menu des jeux
--majm    : met à jour le menu des jeux
--nouv n  : crée le lanceur numéro n
--incr n  : incrémente les numéros de lanceurs >= n
--joue n  : lance le jeu numéro n
```

Écrire le programme principal du script, qui vérifie la présence d'arguments, sinon il affiche l'usage sur la sortie d'erreur puis échoue. Le programme vérifie ensuite la présence du répertoire des lanceurs, au besoin il essaie de le créer. Le programme analyse ensuite les arguments et procède aux actions nécessaires (voir l'usage).

Voici un exemple d'utilisation :

```
./cataga.sh --nouv 2 --incr 7 --nouv 7 --majm --menu --joue 2 --joue 4
```

Voici le code des fonctions pour être complet :

```
# AfficherMenu
#
AfficherMenu ()
{
    if [ -f "$CATAMENU" ]; then
        cat "$CATAMENU"
    else
        echo "Erreur, menu inexistant" 1>&2
    fi
}
```

```

AfficherUsage ()
{
    cat << FIN
    $0 options ...

    Options :
    --help    : affiche cette aide
    --menu    : affiche le menu des jeux
    --majm    : met à jour le menu des jeux
    --nouv n  : crée le lanceur numéro n
    --incr n  : incrémente les numéros de lanceurs >= n
    --joue n  : lance le jeu numéro n
FIN
}

```

Le programme principal demandé est :

```

#!/bin/bash

# Variables globales
CATAREP="./catarep"
CATAMENU="$CATAREP/menu.txt"

# Emplacement des fonctions

# Programme principal

if [ $# -lt 1 ]; then
    AfficherUsage 1>&2 ; exit 1
fi

CreerRepLanceurs

while [ $# -ge 1 ]; do
    case "$1" in
        --help) AfficherUsage ; shift ;;
        --menu) AfficherMenu ; shift ;;
        --majm) ConstruireMenu ; shift ;;
        --nouv) CreerLanceur "${2:-0}" ; shift 2 ;;
        --incr) IncrementerLanceurs "${2:-0}" ; shift 2 ;;
        --joue) LancerJeu "${2:-0}" ; shift 2 ;;
    esac
done

exit 0

```