

# PROGRAMMATION UNIX

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

Ne répondez pas à plusieurs questions en même temps; respectez le découpage de l'énoncé, sous peine de nullité.

## I. Gestion de bloc-notes en bash

On se propose d'écrire un script bash permettant de gérer un bloc-notes. Le bloc-notes est une liste de notes, où chaque note est repérée par son numéro, et est constituée d'un sujet et d'un petit texte.

1) Toutes les notes sont stockées dans le répertoire caché `$HOME/.bloc-notes` de l'utilisateur. Le nom de ce répertoire est mémorisé dans la variable globale `REPNOTES`, initialisée au début du script.

Écrire la fonction `tester_repnotes`. Elle teste l'existence du répertoire `REPNOTES`, au besoin elle le crée en affichant un message explicite. La fonction réussit si le répertoire existe (ou a pu être créé), sinon elle affiche un message d'erreur et échoue.

2) Chaque note est stockée dans un fichier nommé "`bn-x.txt`" dans le répertoire `REPNOTES`, où `x` est le numéro de la note.

Écrire la fonction `nom_note` prenant en argument 1 le numéro d'une note. La fonction affiche sur la sortie standard le chemin complet de la note.

3) Écrire la fonction `trouver_numero` prenant en argument 1 le chemin complet d'une note. La fonction extrait de cet argument le numéro de la note puis l'écrit sur la sortie standard. Cas particulier : si le nom du fichier est `bn-*.txt`, on considère que le numéro est 0.

4) Écrire la fonction `max_numero`. Elle parcourt l'ensemble des notes dans le répertoire `REPNOTES`, cherche le maximum des numéros de notes puis l'affiche sur la sortie standard.

5) Écrire la fonction `ecrire_note` prenant en argument 1 le sujet, puis dans les arguments suivants, les lignes du texte de la note. La fonction cherche le plus grand numéro `x` de note existante dans le répertoire `REPNOTES`, puis crée la note numéro `x+1` : la première ligne est de la forme "`Sujet :  $\square$  sujet`", la seconde ligne est vide, puis viennent les lignes du texte proprement dit.

6) Écrire la fonction `afficher_note` prenant en argument 1 le numéro d'une note. La fonction vérifie que la note existe, sinon elle se termine sans rien faire. Elle affiche ensuite un message indiquant le numéro de la note, puis le contenu du fichier correspondant.

7) Écrire la fonction `afficher_sujet` prenant en argument 1 le numéro `x` d'une note. La fonction vérifie que la note existe, sinon elle se termine sans rien faire. Elle affiche ensuite sur la sortie standard la ligne "`x :  $\square$  sujet`" (en remplaçant bien évidemment `x` et `sujet` par leur valeurs).

8) Écrire la fonction `afficher_sujets`. Elle parcourt l'ensemble des notes du répertoire `REPNOTES` et pour chacune d'entre elles, affiche son sujet au moyen de la fonction précédente.

T.S.V.P.

9) Écrire la fonction `afficher_usage`, affichant au moyen d'un document en ligne (en anglais : *here document*) l'usage suivant du script :

```
Usage: bloc-notes.sh -add sujet ligne1 ligne2 ...
       bloc-notes.sh -list
       bloc-notes.sh -show n
       bloc-notes.sh -del n
```

10) On suppose disposer d'une fonction `supprimer_note`, prenant en argument 1 le numéro d'une note. La fonction vérifie que la note existe, sinon elle se termine sans rien faire. Elle affiche ensuite un message indiquant le numéro de la note, puis supprime le fichier correspondant. *Ne pas écrire cette fonction.*

Écrire le programme principal du script, qui teste la présence d'un argument, sinon affiche l'usage sur la sortie d'erreur puis échoue. Le script teste ensuite la présence du répertoire `REPNOTES`, en le créant au besoin, sinon échoue. Enfin, selon le cas, le script ajoute une note, affiche la liste des sujets, affiche la note numéro *n*, ou supprime la note numéro *n*.

Voici un exemple d'utilisation du script :

```
$ ./blocnotes.sh
Usage: ./blocnotes.sh -add sujet ligne1 ligne2 ...
       ./blocnotes.sh -list
       ./blocnotes.sh -show n
       ./blocnotes.sh -del n

$ ./blocnotes.sh -add "Dates du jury" "L1 : 28 juin" \
"L2 : 29 juin" "L3 : 30 juin"
Création de /home/thiel/.bloc-notes ...
Création de la note /home/thiel/.bloc-notes/bn-1.txt ...

$ ./blocnotes.sh -add "Fin des épreuves" "le 17 juin"
Création de la note /home/thiel/.bloc-notes/bn-2.txt ...

$ ./blocnotes.sh -list
1 : Dates du jury
2 : Fin des épreuves

$ ./blocnotes.sh -show 1
Affichage note 1 ...
Sujet: Dates du jury

L1 : 28 juin
L2 : 29 juin
L3 : 30 juin

$ ./blocnotes.sh -rm 1
Erreur de syntaxe

$ ./blocnotes.sh -del 1
Suppression note 1 ...

$ ./blocnotes.sh -list
2 : Fin des épreuves

$
```

# Correction

## I. Gestion de bloc-notes en bash

On se propose d'écrire un script bash permettant de gérer un bloc-notes. Le bloc-notes est une liste de notes, où chaque note est repérée par son numéro, et est constituée d'un sujet et d'un petit texte.

1) Toutes les notes sont stockées dans le répertoire caché `$HOME/.bloc-notes` de l'utilisateur. Le nom de ce répertoire est mémorisé dans la variable globale `REPNOTES`, initialisée au début du script.

Écrire la fonction `tester_repnotes`. Elle teste l'existence du répertoire `REPNOTES`, au besoin elle le crée en affichant un message explicite. La fonction réussit si le répertoire existe (ou a pu être créé), sinon elle affiche un message d'erreur et échoue.

```
REPNOTES="$HOME/.bloc-notes"

tester_repnotes ()
{
    if [ ! -d "$REPNOTES" ]; then
        echo "Création de $REPNOTES ..."
        if ! mkdir "$REPNOTES" ; then
            echo "Echec création de $REPNOTES" 1>&2
            return 1
        fi
    fi
    return 0
}
```

2) Chaque note est stockée dans un fichier nommé `"bn-x.txt"` dans le répertoire `REPNOTES`, où *x* est le numéro de la note.

Écrire la fonction `nom_note` prenant en argument 1 le numéro d'une note. La fonction affiche sur la sortie standard le chemin complet de la note.

```
nom_note () # num
{
    echo "$REPNOTES/bn-$1.txt"
}
```

3) Écrire la fonction `trouver_numero` prenant en argument 1 le chemin complet d'une note. La fonction extrait de cet argument le numéro de la note puis l'écrit sur la sortie standard. Cas particulier : si le nom du fichier est `bn-*.txt`, on considère que le numéro est 0.

```
trouver_numero () # chemin/...../bn-123.txt
{
    local n f

    f="${1##*/bn-}"
    n="${f%.txt}"
    if [ "$n" = "*" ]; then n="0" ; fi
    echo "$n"
}
```

4) Écrire la fonction `max_numero`. Elle parcourt l'ensemble des notes dans le répertoire `REPNOTES`, cherche le maximum des numéros de notes puis l'affiche sur la sortie standard.

```

max_numero ()
{
    local n f max=0

    for f in "$REPNOTES"/bn-*.txt
    do
        n=$(trouver_numero "$f")
        if ((n > max)); then max="$n" ; fi
    done
    echo "$max"
}

```

5) Écrire la fonction `ecrire_note` prenant en argument 1 le sujet, puis dans les arguments suivants, les lignes du texte de la note. La fonction cherche le plus grand numéro  $x$  de note existante dans le répertoire `REPNOTES`, puis crée la note numéro  $x+1$  : la première ligne est de la forme "`Sujet:  $\square$ sujet`", la seconde ligne est vide, puis viennent les lignes du texte proprement dit.

```

ecrire_note () # sujet ligne1 ligne2 ...
{
    local n f ligne

    n=$(max_numero) ; ((n++))
    f=$(nom_note $n)
    echo "Création de la note $f ..."

    echo "Sujet: $1" >| "$f"
    echo "" >> "$f"
    shift
    for ligne ; do
        echo "$ligne" >> "$f"
    done
}

```

6) Écrire la fonction `afficher_note` prenant en argument 1 le numéro d'une note. La fonction vérifie que la note existe, sinon elle se termine sans rien faire. Elle affiche ensuite un message indiquant le numéro de la note, puis le contenu du fichier correspondant.

```

afficher_note () # num
{
    local f

    f=$(nom_note "$1")
    if [ ! -f "$f" ]; then return ; fi
    echo "Affichage note $1 ..."
    cat "$f"
}

```

7) Écrire la fonction `afficher_sujet` prenant en argument 1 le numéro  $x$  d'une note. La fonction vérifie que la note existe, sinon elle se termine sans rien faire. Elle affiche ensuite sur la sortie standard la ligne " `$x$ :  $\square$ sujet`" (en remplaçant bien évidemment  $x$  et *sujet* par leur valeurs).

```

afficher_sujet () # num
{
    local f mot1 sujet

    f=$(nom_note "$1")
    if [ ! -f "$f" ]; then return ; fi
    read mot1 sujet < "$f"
    echo "$1 : $sujet"
}

```

8) Écrire la fonction `afficher_sujets`. Elle parcourt l'ensemble des notes du répertoire `REPNOTES` et pour chacune d'entre elles, affiche son sujet au moyen de la fonction précédente.

```
afficher_sujets ()
{
    local n f

    for f in "$REPNOTES"/bn-*.txt
    do
        n=$(trouver_numero "$f")
        afficher_sujet "$n"
    done
}
```

9) Écrire la fonction `afficher_usage`, affichant au moyen d'un document en ligne (en anglais : *here document*) l'usage suivant du script :

```
Usage: bloc-notes.sh -add sujet ligne1 ligne2 ...
       bloc-notes.sh -list
       bloc-notes.sh -show n
       bloc-notes.sh -del n

afficher_usage ()
{
    cat << EOT
Usage: $0 -add sujet ligne1 ligne2 ...
       $0 -list
       $0 -show n
       $0 -del n
EOT
}
```

10) On suppose disposer d'une fonction `supprimer_note`, prenant en argument 1 le numéro d'une note. La fonction vérifie que la note existe, sinon elle se termine sans rien faire. Elle affiche ensuite un message indiquant le numéro de la note, puis supprime le fichier correspondant. *Ne pas écrire cette fonction.*

Écrire le programme principal du script, qui teste la présence d'un argument, sinon affiche l'usage sur la sortie d'erreur puis échoue. Le script teste ensuite la présence du répertoire `REPNOTES`, en le créant au besoin, sinon échoue. Enfin, selon le cas, le script ajoute une note, affiche la liste des sujets, affiche la note numéro `n`, ou supprime la note numéro `n`.

```
#!/bin/bash

# programme principal

if [ $# -eq 0 ]; then
    afficher_usage 1>&2
    exit 1
fi

if ! tester_repnotes ; then exit 1 ; fi

case "$1" in
    -help) afficher_usage ;;
    -add) shift ; ecrire_note "$@" ;;
    -list) afficher_sujets ;;
    -show) afficher_note "${2:-0}" ;;
    -del) supprimer_note "${2:-0}" ;;
    *) echo "Erreur de syntaxe" 1>&2 ; exit 1 ;;
esac

exit 0
```

Pour être complet, voici le code des fonctions manquantes :

```
supprimer_note () # num
{
    local f

    f=$(nom_note "$1")
    if [ ! -f "$f" ]; then return ; fi
    echo "Suppression note $1 ..."
    rm -f "$f"
}
```