Corrigé du TP 8 Système

C. Pain-Barre

INFO - IUT Aix-en-Provence

version du 12/12/2012

---

1 Utilisation des boucles

Corrigé de l'exercice 1

Se familiariser avec l'utilisation de la boucle **while** *avec un test d'expression arithmétique. L'exercice est complété par l'utilisation de l'instruction* **if** *dans le corps de la boucle.*

1. `$ var=10`
2.
```
$ while ((var > 0)); do
> ((var--))
> echo "J'en ai enlevé un et var vaut $var"
> done
J'en ai enlevé un et var vaut 9
...
J'en ai enlevé un et var vaut 0
$
```
   ➪ pour plus de concision, en sachant que **var** est positive au départ, les deux premières lignes peuvent être remplacées par la seule ligne :
   ```
   while ((var--)); do
   ```
3.
```
$ var=10
$ while ((var > 0)); do
> ((var--))
> echo -n "J'en ai enlevé un et var vaut $var"
> if ((var % 2 == 0)); then
> echo " (paire)"
> else
> echo " (impaire)"
> fi
> done
J'en ai enlevé un et var vaut 9 (impaire)
...
J'en ai enlevé un et var vaut 0 (paire)
```
4. `$ unset var`

Corrigé de l'exercice 2 (attente active)

Utiliser une boucle **while** *pour réaliser une attente active.*

1. ```
   temps_total=0
   echo -n "attente de la création de fictest"
   while ! [ -f fictest ]; do
     sleep 1
     echo -n "."
     ((temps_total++))
   done
   echo -e "\nsecondes attendues : $temps_total"
   ```
2. *. . . pas besoin de corrigé pour cette question. . .*

Corrigé de l'exercice 3

Utilisation des boucles **for**

1. ```
   $ tab=(zero un deux trois quatre cinq six sept)
   ```
2. ```
   $ for ((i = 0; i < ${#tab[@]}; i += 2)); do
   > echo "${tab[i]}"
   > done
   zero
   ...
   six
   ```
3. ```
   $ afficher=0                    # afficher élément (faux)
   $ for elt in "${tab[@]}"
   > do
   > if ((afficher)); then
   > echo "$elt"
   > fi
   > ((afficher = ! afficher))     # ou   let afficher=1-afficher
   > done
   un
   ...
   sept
   ```
4. ```
   $ unset tab
   ```

Corrigé de l'exercice 4

Utilisation de **for** *et des modificateurs de substitution de variables*

1. ```
   $ cd ~/tp ; cp -rv tpunix testfor
   ```
2. ```
   $ cd testfor
   ```
3. ```
   $ for fic in *.txt; do
   > mv "$fic" "$fic.old"
   > echo "fichier '$fic' renommé en fichier '$fic.old'"
   > done
   fichier 'acrostiche.txt' renommé en 'acrostiche.txt.old'
   ```

```
fichier 'amphigouri.txt' renommé en 'amphigouri.txt.old'
...
$ ls     # pour vérifier
```

4.
```
$ for fic in *.txt.old; do
> mv -v "$f" "${f%.old}"
> done
`acrostiche.txt.old' -> `acrostiche.txt'
`amphigouri.txt.old' -> `amphigouri.txt'
`amphi.txt.old' -> `amphi.txt'
`cigale.txt.old' -> `cigale.txt'
`simpsons.txt.old' -> `simpsons.txt'
...
$ ls     # pour vérifier
```

5.
```
$ fics=(*.txt)   # création du tableau fics contenant les fichiers
$ for ((i = 0; i < ${#fics[@]}; i++)); do
> fic="${fics[i]}"
> mv "$fic" "$fic.old"
> echo "fichier '$fic' renommé en '$fic.old'"
> done
fichier 'acrostiche.txt' renommé en 'acrostiche.txt.old'
fichier 'amphigouri.txt' renommé en 'amphigouri.txt.old'
fichier 'amphi.txt' renommé en 'amphi.txt.old'
fichier 'cigale.txt' renommé en 'cigale.txt.old'
fichier 'simpsons.txt' renommé en 'simpsons.txt.old'
...
$ ls     # pour vérifier
```

6.
```
$ fics=(*.txt.old)   # création du tableau fics contenant les fichiers
$ for ((i = 0; i < ${#fics[@]}; i++)); do
> fic="${fics[i]}"
> mv -v "$fic" "${fic%.old}"
> done
`acrostiche.txt.old' -> `acrostiche.txt'
`amphigouri.txt.old' -> `amphigouri.txt'
`amphi.txt.old' -> `amphi.txt'
`cigale.txt.old' -> `cigale.txt'
`simpsons.txt.old' -> `simpsons.txt'
...
$ ls     # pour vérifier
```

7.
```
$ cd ../tpunix ; rm -rf ../testfor
```

Corrigé de l'exercice 5

Normaliser le nom de plusieurs fichiers en utilisant une boucle **for**, *les substitutions de variables avec motifs et* **mv**

1. `$ cp -r ~cpb/public/unix/fics-espaces .`
2. `$ cd fics-espaces ; ls`
3.
```
$ for fic in *; do
> newfic=${fic//␣/_}
> newfic=${newfic//\%20/_}
> if [ "$fic" != "$newfic" ]; then
> mv -v "$fic" "$newfic"
> else
> echo "le fichier '$fic' est déjà normalisé"
> fi
> done
`Albert%20Einstein.txt' -> `Albert_Einstein.txt'
`Asterix le Gaulois.txt' -> `Asterix_le_Gaulois.txt'
...
```

Corrigé de l'exercice 6

Familiarisation avec la boucle **select**

1. *... pas besoin de corrigé pour cette question...*
2.
```
1) afficher RANDOM
2) quitter
#? 1
vous avez saisi : 1
RANDOM = 8282
#? 4
vous avez saisi : 4
mauvais choix !
#? 1
vous avez saisi : 1
RANDOM = 583
#? quitter
vous avez saisi : quitter
mauvais choix !
#? 2
vous avez saisi : 2
bye bye
$
```

2 Écriture de scripts

Corrigé de l'exercice 7

Script qui normalise le nom des fichiers contenus dans des répertoires passés en arguments.

```
#!/bin/bash

# Script : normrep.bash

function normfic {
    local fic=$1
    local newfic=${fic// /_}
    newfic=${newfic//\%20/_}

    if [ "$fic" != "$newfic" ]; then
        mv -v "$fic" "$newfic"
    else
        echo "le fichier '$fic' est déjà normalisé"
    fi
}

function trtrep {
    local rep=$1

    if ! [ -r "$rep" -a -w "$rep" -a -x "$rep" ]; then
      echo "droits insuffisants pour traiter '$rep'" >&2
      return 1
    fi

    local repcourant=$PWD

    echo "traitement de $rep"
    cd "$rep"
    for fic in *; do
        normfic "$fic"
    done

    cd "$repcourant"
}

reps=(.)
(($# != 0)) && reps=("$@")

retour=0
for rep in "${reps[@]}"
do
  trtrep "$rep" || retour=1
done
exit $retour
```

Corrigé de l'exercice 8

Scripts affichant un menu proposant des actions à réaliser. Cet exercice illustre l'emploi de **select**, *ainsi que de* **test** *pour vérifier des conditions sur les fichiers/répertoires.*

1.
```
#!/bin/bash

# Script : menu1.bash

PS3="Choix ? "
select choix in "afficher le répertoire courant"             \
                "se déplacer dans un répertoire"             \
                "afficher le contenu du répertoire courant"  \
                "afficher le contenu d'un répertoire"        \
                "quitter"
do
  case "$choix" in
    "afficher le répertoire de travail" )
        pwd
        ;;

    "se déplacer dans un répertoire" )
        read -p "répertoire ? " rep
        cd "$rep"
        ;;

    "afficher le contenu détaillé du répertoire courant" )
        ls -l
        ;;

    "afficher le contenu détaillé d'un répertoire" )
        read -p "répertoire ? " rep
        ls -l "$rep"
        ;;

    "quitter" )
        break
        ;;

    * )
        echo "Saisie incorrecte" >&2
        ;;
  esac
done
```


2.
```
#!/bin/bash

# Script : menu2.bash

PS3="Choix ? "
select choix in "afficher le répertoire de travail"                    \
                "se déplacer dans un répertoire"                       \
                "afficher le contenu détaillé du répertoire courant"   \
                "afficher le contenu détaillé d'un répertoire"         \
                "quitter"
do
  case "$choix" in
    "afficher le répertoire de travail" )
        pwd
        ;;

    "se déplacer dans un répertoire" )
        read -p "répertoire ? " rep
        if [ -d "$rep" ]; then
          if [ -x "$rep" ]; then
            cd "$rep"
          else
            echo "droits insuffisants pour accéder à $rep" >&2
          fi
        else
          echo "$rep est inexistant" >&2
        fi
        ;;

    "afficher le contenu détaillé du répertoire courant" )
        if [ -r . ]; then
          ls -l
        else
          echo "impossible de lire le contenu du répertoire de travail" >&2
        fi
        ;;

    "afficher le contenu détaillé d'un répertoire" )
        read -p "répertoire ? " rep
        if [ -d "$rep" ]; then
          if [ -x "$rep" -a -r "$rep" ]; then
            ls -l "$rep"
          else
            echo "droits insuffisants pour l'affichage détaillé de $rep" >&2
          fi
        else
          echo "$rep est inexistant" >&2
        fi
        ;;

    "quitter" )
        break
        ;;

    * )
        echo "Saisie incorrecte" >&2
        ;;
  esac
done
```

Corrigé de l'exercice 9

Tri de nombres en arguments et liens physiques

1.
```
#!/bin/bash

# Script : sortup.bash

# il est plus simple de passer par un tableau...
tab=("$@")

# qu'on va trier (on évite de le recréer) par un tri classique
# par sélection.

for ((i = 0; i < ${#tab[*]} - 1; i++))
do
    indmin=$i
    for ((j = i+1; j < ${#tab[*]}; j++))
    do
        if ((${tab[j]} < ${tab[indmin]})); then
            indmin=$j
        fi
    done

    # permutation élément i et indmin
    tmp="${tab[i]}"
    tab[i]="${tab[indmin]}"
    tab[indmin]="$tmp"
done

echo "${tab[@]}" # remarque : écrit une ligne vide si aucun élément à trier...
```
2.
```
ln sortup sortdn
```

3. Il s'agit juste de modifier l'opérande de comparaison des éléments (qui devient **>** pour **sortdn.bash**). Cette opérande est stockée dans la variable **comparaison**. Cependant, bash est un peu capricieux, et il faut utiliser **$comparaison** (et pas seulement **comparaison**) dans le test de la condition arithmétique.

```
#!/bin/bash

# Script : sortup.bash ou sortdn.bash

if [ ${0##*/} == sortup.bash ]; then
    comparaison="<"       # ordre croissant
else
    comparaison=">"       # ordre décroissant
fi

# il est plus simple de passer par un tableau...
tab=("$@")

# qu'on va trier (on évite de le recréer) par un tri classique
# par sélection.

for ((i = 0; i < ${#tab[*]} - 1; i++))
do
    indmin=$i
    for ((j = i+1; j < ${#tab[*]}; j++))
    do
        if ((${tab[j]} $comparaison ${tab[indmin]})); then
            indmin=$j
        fi
    done

    # permutation élément i et indmin
    tmp="${tab[i]}"
    tab[i]="${tab[indmin]}"
    tab[indmin]="$tmp"
done

echo "${tab[@]}" # remarque : écrit une ligne vide si aucun élément à trier...
```

Corrigé de l'exercice 10

Script gérant un compte bancaire

```
#!/bin/bash

# Script : gestion.bash

if (($# != 2)); then
    echo "Usage : ${0##*/} (-c|-d) montant"
    exit 1
fi

ficsolde="$HOME/tp/tpunix/solde.txt"
ficopers="$HOME/tp/tpunix/operations.txt"

case "$1" in
    -c )
        operation="credit"
        operateur=+
        ;;

    -d )
        operation="débit"
        operateur=-
        ;;

    * )  # erreur
        echo "Usage : ${0##*/} (-c|-d) montant"
        exit 1
        ;;
esac

read -p "libellé de l'opération ? " libelle

echo -e "$(date '+%x')\t$operation\t${2}\t$libelle" >> "$ficopers"

if [ -f "$ficsolde" ]; then
    solde="$(cat "$ficsolde")"
else
    solde=0
fi

echo $((solde $operateur $2)) > "$ficsolde"
```

Corrigé de l'exercice 11

Gestion de fichiers musicaux au format MP3

1. *... pas besoin de corrigé pour cette question...*
2.
```
#!/bin/bash

function get_auteur {
    id3v2 -l "$1" | sed -n -e 's/^TPE1[^:]*: //p'
}

function get_titre {
    id3v2 -l "$1" | sed -n -e 's/^TIT2[^:]*: //p'
}

function get_annee {
    id3v2 -l "$1" | sed -n -e 's/^TYER[^:]*: //p'
}

function get_album {
    id3v2 -l "$1" | sed -n -e 's/^TALB[^:]*: //p'
}

function get_piste {
    id3v2 -l "$1" | sed -n -e 's/^TRCK[^:]*: //p'
}

function set_auteur {
    id3v2 --TPE1 "$2" "$1"
}

function set_titre {
    id3v2 --TIT2 "$2" "$1"
}

function set_annee {
    id3v2 --TYER "$2" "$1"
}

function set_album {
    id3v2 --TALB "$2" "$1"
}

function set_piste {
    id3v2 --TRCK "$2" "$1"
}

if (($# != 1)); then
    echo "un (seul) argument attendu" >> /dev/stderr
    exit 2
fi

if ! [ -f "$1" -a -r "$1" ]; then
    echo "fichier '$1' inexistant ou inaccessible" >> /dev/stderr
    exit 1
fi

menu=("Afficher l'auteur" \
      "Afficher le titre" \
      "Afficher l'album"  \
      "Afficher l'année"  \
      "Afficher le numéro de piste" \
      "Modifier/Ajouter l'auteur" \
      "Modifier/Ajouter le titre" \
```

```
        "Modifier/Ajouter l'album" \
        "Modifier/Ajouter l'année" \
        "Modifier/Ajouter la piste" \
        "Quitter")
    PS3="Que voulez-vous faire sur '$1' ? "

    select choix in "${menu[@]}"; do
        case "$choix" in
            "Afficher l'auteur" )
                get_auteur "$1"
                ;;

            "Afficher le titre" )
                get_titre "$1"
                ;;

            "Afficher l'album" )
                get_album "$1"
                ;;

            "Afficher l'année" )
                get_annee "$1"
                ;;

            "Afficher le numéro de piste" )
                get_piste "$1"
                ;;

            "Modifier/Ajouter l'auteur" )
                if ! [ -w "$1" ]; then
                    echo "Impossible : fichier en lecture seule" >> /dev/stderr
                else
                    echo "Auteur actuel : $(get_auteur "$1")"
                    read -e -p 'Nouvel auteur ? ' auteur
                    set_auteur "$1" "$auteur"
                fi
                ;;

            "Modifier/Ajouter le titre" )
                if ! [ -w "$1" ]; then
                    echo "Impossible : fichier en lecture seule" >> /dev/stderr
                else
                    echo "Titre actuel : $(get_titre "$1")"
                    read -e -p 'Nouveau titre ? ' titre
                    set_titre "$1" "$titre"
                fi
                ;;

            "Modifier/Ajouter l'album" )
                if ! [ -w "$1" ]; then
                    echo "Impossible : fichier en lecture seule" >> /dev/stderr
                else
                    echo "Album actuel : $(get_album "$1")"
                    read -e -p 'Nouvel album ? ' album
                    set_album "$1" "$album"
                fi
                ;;

            "Modifier/Ajouter l'année" )
                if ! [ -w "$1" ]; then
                    echo "Impossible : fichier en lecture seule" >> /dev/stderr
                else
                    echo "Année actuelle : $(get_annee "$1")"
                    read -e -p 'Nouvelle année ? ' annee
                    if ! echo "$annee" | grep -q '^[0-9]\+$'; then
                        echo "erreur : '$annee' n'est pas un nombre" >> /dev/stderr
```

```
                else
                    set_annee "$1" "$annee"
                fi
            fi
            ;;

        "Modifier/Ajouter la piste" )
            if ! [ -w "$1" ]; then
                echo "Impossible : fichier en lecture seule" >> /dev/stderr
            else
                echo "Piste actuelle : $(get_piste "$1")"
                read -e -p 'Nouvelle piste ? ' piste
                if ! echo "$piste" | grep -q '^[0-9]\+$'; then
                    echo "erreur : '$piste' n'est pas un nombre" >> /dev/stderr
                else
                    set_piste "$1" "$piste"
                fi
            fi
            ;;

        "Quitter" )
            exit 0
        ;;

        * )
            echo "choix incorrect" >> /dev/stderr
            ;;
    esac
done
```

3.
```
#!/bin/bash

DIRCOL="$HOME/musique"

# emplacement des définitions des fonctions (voir tagedit.bash)

mkdir -p "$DIRCOL"

for nomfic in "$1"/*; do
    if ! echo "$nomfic" | grep -q -i  "\.mp3$"; then
        echo "fichier '$nomfic' ignore" >> /dev/stderr
        continue
    fi

    auteur="$(get_auteur "$nomfic")"
    album="$(get_album "$nomfic")"
    annee="$(get_annee "$nomfic")"
    titre="$(get_titre "$nomfic")"
    piste="$(get_piste "$nomfic")"
    if ((${#piste} < 2)); then
        piste="0$piste"
    fi

    if ! [ -d "$DIRCOL/$auteur" ]; then
        echo "Nouvel auteur : $auteur"
        mkdir "$DIRCOL/$auteur"
    fi

    if ! [ -d "$DIRCOL/$auteur/$annee-$album" ] ; then
        echo "Nouvel album de '$auteur' : $album"
        mkdir "$DIRCOL/$auteur/$annee-$album"
    fi

    mv "$nomfic" "$DIRCOL/$auteur/$annee-$album/$piste-$titre.mp3"

done
```

Corrigé de l'exercice 12

Gestion d'une bibliothèque

1.
```
#!/bin/bash

# Script : inscrire.bash

function usage {
    echo "Usages : ${0##*/} -a nom prénom adresse" >&2
    echo "         ${0##*/} -l exemplaires titre auteur" >&2
}

function inscrire_adherent {
    if [ -f membres ]; then
        derniernum="$(tail -1 membres | cut -d ';' -f 1)"
    else
        derniernum=0
    fi
    echo "$((derniernum + 1));$1;$2;$3" >> membres
}

function inscrire_livre {
    if [ -f livres ]; then
        derniernum="$(tail -1 livres | cut -d ';' -f 1)"
    else
        derniernum=0
    fi
    echo "$((derniernum + 1));$2;$3" >> livres
    for ((i = 1; i <= $1; i++)) do
        echo "$((derniernum + 1));$i;oui" >> exemplaires
    done
}

case "$1" in
    -a )
        if (($# != 4)); then
            usage
            exit 1
        fi
        inscrire_adherent "${@:2}"
        ;;

    -l )
        if (($# != 4)); then
            usage
            exit 1
        fi
        inscrire_livre "${@:2}"
        ;;

    * )
        usage
        exit 1
        ;;
esac
```

2.
```
#!/bin/bash

# Script : demande.bash

if (($# != 2)); then
    echo "Usage : ${0##*/} adherent titre" >&2
    exit 1
fi

adherent="$1"
titre="$2"

numadher="$(grep "^[^;]*;$adherent;" membres | cut -d ';' -f 1)"
if [ -z "$numadher" ]; then
    echo "$adherent inexistant" >&2
    exit 1
fi

numlivre="$(grep "^[^;]*;$titre;" livres | cut -d ';' -f 1)"
if [ -z "$numlivre" ]; then
    echo "$titre inexistant" >&2
    exit 1
fi

# on ne veut qu'un seul exemplaire disponible :
# l'option -m de grep fixe un nombre max de lignes recherchées
# on aurait aussi pu utiliser head pour ne garder que la première
exemplaire="$(grep -m 1 "^$numlivre;.*;oui" exemplaires  | cut -d';' -f 2)"
if [ -z "$exemplaire" ] ; then
    echo "aucun exemplaire disponible"
    exit 1
fi

echo "$numadher;$numlivre;$exemplaire;$(date '+%-d %-m %Y %X')" >> emprunts

sed -i -e "/^$numlivre;$exemplaire;/s/oui/non/" exemplaires

echo "emprunt du livre enregistré"
```

3.
```
#!/bin/bash

# Script : rend.bash

if (($# != 2)); then
    echo "Usage : ${0##*/} adherent titre" >&2
    exit 1
fi

adherent="$1"
titre="$2"

numadher="$(grep "^[^;]*;$adherent;" membres | cut -d ';' -f 1)"
if [ -z "$numadher" ]; then
    echo "$adherent inexistant" >&2
    exit 1
fi

numlivre="$(grep "^[^;]*;$titre;" livres | cut -d ';' -f 1)"
if [ -z "$numlivre" ]; then
    echo "$titre inexistant" >&2
    exit 1
fi

exemprunt="$(grep -m 1 "^$numadher;$numlivre;" emprunts | cut -d';' -f 3)"
if [ -z "$exemprunt" ]; then
    echo "$adherent n'a pas emprunté $titre"
    exit 1
fi

grep -v "^$numadher;$numlivre;$exemprunt;" emprunts > emprunts.tmp
mv emprunts.tmp emprunts

sed -i -e "/^$numlivre;$exemprunt;/s/non/oui/" exemplaires

echo "rendu du livre enregistré"
```

4.
```
#!/bin/bash

# Script : comptemps.bash


if (($# != 4)); then
    echo "Usage : ${0##*/} jour mois année heures:minutes:secondes"
    exit 1
fi


date_crt=($(date '+%-d %-m %Y %X'))

# normalisation en nombre de jours depuis 00/00/0000
norm_arg=$(($3*365 + $2*30 + $1))
norm_crt=$((${date_crt[2]}*365 + ${date_crt[1]}*30 + ${date_crt[0]}))

if ((norm_arg + 365 <= norm_crt)); then
    exit 3
elif ((norm_arg + 30 <= norm_crt)); then
    exit 2
elif ((norm_arg + 1 <= norm_crt)); then
    exit 1
else
    exit 0
fi
```

5.
```
#!/bin/bash

# Script : rappel.bash

while read ligne;
do
    ./temps.bash $(echo "$ligne" | cut -d';' -f 4)
    if (($? >= 2)); then
        echo "$ligne"
    fi
done < emprunts
```

Corrigé de l'exercice 13

Le pendu — 1re version

```
#!/bin/bash

# Script : pendu1.bash
# version travaillant directement sur les chaines, comme sur un tableau

nbmots=$(cat mots | wc -l)

nummot=$((1 + RANDOM % nbmots))             # tirage au sort du mot

mot=$(head -n $nummot mots | tail -n 1)     # récupération du mot

# fabrication du mot masqué
masque="${mot:0:1}"                         # première lettre du mot
for ((i = 1; i < ${#mot} - 1; i++)); do     # puis on remplit
    masque="$masque."                       # l'intérieur avec des
done                                        # points
masque="$masque${mot: -1}"                  # puis la dernière lettre


nb_a_trouver=$((${#mot} -2))                # nombre de lettres à trouver

nb_essais=7
while ((nb_a_trouver)); do
    echo -n "$masque"

    read -p " : lettre ? " lettre

    trouve=0
    for ((i = 0; i < ${#mot}; i++)); do          # recherche de la
        if [ "$lettre" == "${mot:i:1}" ]; then   # lettre entrée
            trouve=1
            if [ "${masque:i:1}" == "." ]; then
                masque="${masque:0:i}${mot:i:1}${masque:i+1}"
                ((--nb_a_trouver))
            fi
        fi
    done

    if (( ! trouve )); then
        if ((--nb_essais < 0)); then
            echo "Perdu !"
            echo "Le mot était : $mot"
            exit 1
        fi
        echo "Erreur : plus que $nb_essais erreurs permises"
    fi
done

echo "Gagné : $mot"
```

Le pendu — 2e version

```
#!/bin/bash

# Script : pendu2.bash
# version travaillant sur des tableaux

nbmots=$(cat mots | wc -l)

nummot=$((1 + RANDOM % nbmots))              # tirage au sort du mot

mot=$(head -n $nummot mots | tail -n 1)      # récupération du mot

# transformation du mot en tableau
lettres=($(echo $mot | sed 's/./& /g'))

masque=(${lettres[@]})                              # copie dans masque en
for ((i = 1; i < ${#masque[*]} - 1; i++)); do       # remplaçant les lettres
    masque[i]='.'                                   # à trouver par '.'
done

nb_a_trouver=$((${#mot} -2))                 # nombre de lettres à trouver

nb_essais=7
while ((nb_a_trouver)); do
                                             # écriture du mot masqué en modifiant
    ( IFS='' ; echo -n "${masque[*]}" )      # IFS dans un sous-shell

    read -p " : lettre ? " lettre

    trouve=0
    for ((i = 0; i < ${#lettres[*]}; i++)); do      # recherche de la
        if [ "$lettre" == "${lettres[i]}" ]; then   # lettre entrée
            trouve=1
            if [ "${masque[i]}" == "." ]; then
                masque[i]=${lettres[i]}
                ((--nb_a_trouver))
            fi
        fi
    done

    if (( ! trouve )); then
        if ((--nb_essais < 0)); then
            echo "Perdu !"
            echo "Le mot était : $mot"
            exit 1
        fi
        echo "Erreur : plus que $nb_essais erreurs permises"
    fi
done

echo "Gagné : $mot"
```

Corrigé de l'exercice 14

Scripts gérant un carnet d'adresses

```
#!/bin/bash

# Script : phone4.bash

###############################################################################
#
# Définition de quelques constantes pratiques
#
###############################################################################

#
# référence du répertoire téléphonique (pour faciliter les changements) :
#
reptel=${HOME}/tp/tpunix/telephones.txt

# remarquons que pour l'utilisateur toto de 1ere année, la variable
# reptel contient déjà /users/etud1/toto/tp/tpunix/telephones.txt

#
# Codes de retour :
#
fait=0          # si l'action a été réalisée avec succès
echec=1         # si l'action est infructueuse
erreur=2        # en cas de mauvaise utilisation


tab_choix=(Ajouter    \
           Afficher   \
           Rechercher \
           Supprimer  \
           Modifier   \
           Quitter)

tab_fonc_menu=(lire_ajouter_contact    \
               afficher_repertoire     \
               lire_rechercher_contact \
               lire_supprimer_contact  \
               lire_modifier_contact   \
               quitter)

#
# Messages informant sur le répertoire :
#
no_fic="le répertoire est inacessible."
vide="le répertoire est vide."

###############################################################################
#
# Fonctions diverses
#
###############################################################################

function ecrire_aide {

    echo "${0##*/} -h : affiche cette aide"
    echo "${0##*/} -a nom prénom infos : ajoute un contact"
    echo "${0##*/} -l : affiche le répertoire"
    echo "${0##*/} -d nom : supprimer des contacts"
    echo "${0##*/} -u nom nouvnom nouvprenom nouvinfos : modifier des contacts"
    echo "${0##*/} nom : rechercher des contacts"
```

```
}

function erreur {

    echo -e "$@" >&2

}

#############################################################################
#
# Fonctions de gestion du répertoire
#
#############################################################################

function ajouter_contact {

    echo -e "$1\t$2\t${@:3}" >> $reptel;
    echo "contact ajouté"
}

function afficher_repertoire {

    if [ ! -f $reptel ]; then
        erreur "Affichage impossible : $no_fic\n"
        exit $echec
    fi

    echo -e "Affichage du répertoire :\n"

    if [ -s $reptel ]; then
        less $reptel
        echo -e "Terminé\n"
    else
        echo -e "Vide\n"
    fi

}

function rechercher_contact {

    if [ ! -f $reptel ]; then
        erreur "Recherche impossible : $no_fic\n"
        exit $echec
    fi

    echo  "Elements trouvés :"

    if grep -q "^$1" $reptel; then
        grep "^$1" $reptel | less
    else
        echo "    Aucun"
        exit $echec
    fi

}

function supprimer_contact {

    if [ ! -f $reptel ]; then
        erreur "Recherche impossible : $no_fic\n"
        exit $echec
    fi

    echo "Eléments supprimés : "

    if grep -qw "^$1" $reptel; then
```

```
        grep -w "^$1" $reptel | less
        grep -vw "^$1" $reptel > ${reptel}.tmp
        mv ${reptel}.tmp $reptel
        echo "Suppression réalisée"
    else
        echo "    Aucun"
        exit $echec
    fi

}

function modifier_contact {

    if [ ! -f $reptel ]; then
        erreur "Modification impossible : $no_fic\n"
        exit $echec
    fi

    echo "Eléments modifiés : "

    if grep -qw "^$1" $reptel; then
        grep -w "^$1" $reptel | less
        grep -vw "^$1" $reptel > ${reptel}.tmp
        mv ${reptel}.tmp $reptel
        echo -e "$2\t$3\t$4" >> $reptel
        echo "Modification effectuée"
    else
        echo "    Aucun"
        exit $echec
    fi
}

function quitter {

    exit $fait

}


################################################################################
#
# Fonctions de lecture pour choix par menu
#
################################################################################


function lire_ajouter_contact {

    read -p "nom ? " nom
    read -p "prenom ? " prenom
    read -p "infos ? " infos

    ajouter_contact "$nom" "$prenom" "$infos"

}

function lire_rechercher_contact {

    read -p "nom à rechercher ? " chaine

    rechercher_contact "$chaine"

}

function lire_supprimer_contact {
```

```
    read -p "nom à supprimer ? " nom

    supprimer_contact "$nom"
}

function lire_modifier_contact {

    read -p "nom à modifier ? " chaine
    read -p "nouveau nom ? " nom
    read -p "nouveau prénom ? " prenom
    read -p "nouvelles infos ? " infos

    modifier_contact "$chaine" "$nom" "$prenom" "$infos"

}

###############################################################################
#
# Détermination des choix selon menu ou arguments
# et vérification des arguments
#
###############################################################################

if (($# == 0)); then       # le choix se fait par menu

    erreur "\nGestion du répertoire téléphonique"
    erreur "---------------------------------\n"

    PS3="votre choix ? "
    select choix in "${tab_choix[@]}"
    do
        if [ ! "$choix" ]; then
            erreur "choix invalide"
        else
            ${tab_fonc_menu[REPLY-1]}
        fi
    done

else                # le choix se fait par les arguments

    case "$1" in

        -h )
            ecrire_aide
            exit $fait
            ;;

        -a )
            if (($# < 4)); then
                erreur "${0##*/}: arguments manquants"
                erreur "${0##*/} -h pour obtenir de l'aide"
                exit $erreur
            fi

            ajouter_contact "${@:2}"

            exit $fait
            ;;

        -l )
            if (($# != 1)); then
                erreur "${0##*/}: trop d'arguments"
                erreur "${0##*/} -h pour obtenir de l'aide"
                exit $erreur
            fi
```

```
            afficher_repertoire

            exit $fait
            ;;

        -d )
            if (($# == 1)); then
                erreur "${0##*/}: pas assez d'arguments"
                erreur "${0##*/} -h pour obtenir de l'aide"
                exit $erreur
            fi

            supprimer_contact "${*:2}"

            exit $fait
            ;;

        -u )
            if (($# < 5)); then
                erreur "${0##*/}: pas assez d'arguments"
                erreur "${0##*/} -h pour obtenir de l'aide"
                exit $erreur
            fi

            modifier_contact "$2" "$3" "$4" "${*:5}"

            exit $fait
            ;;

        -* )
            erreur "${0##*/}: $1 option non reconnue"
            erreur "${0##*/} -h pour obtenir de l'aide"
            exit $erreur
            ;;

        *)
            rechercher_contact "$*"

            exit $fait
            ;;

    esac

fi
```