Projet NSY103 linux

Auditeur : Rachik FETTACHE

2018 - 2019

Les objectifs

- Premier paramètre : à partir d'un fichier dictionnaire dico.txt (en majuscule)

- Statistiques sur l'utilisation des lettres dans une langue

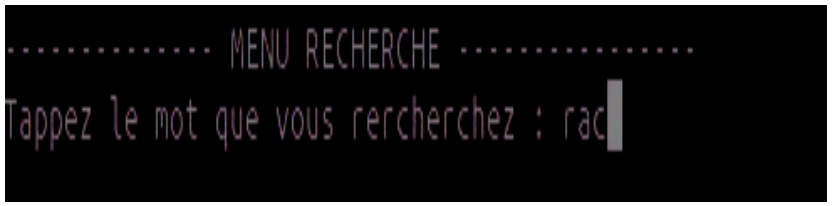
Faire un classement sur l'utilisation des lettres

```
STATISTIQUES LETTRES DANS LE DICTIONNAIRE :
278814    -    E
229938    -    A
219131    -    I
210391    -    R
207889    -    S
179165    -    N
176030    -    T
158282    -    O
108195    -    U
106300    -    L
98832     -    C
74421     -    M
72481     -    D
69141     -    P
48942     -    G
43471     -    B
36777     -    F
35940     -    H
34982     -    Z
30562     -    V
16340     -    Q
11030     -    Y
8262      -    X
5743      -    J
1595      -    K
529       -    W
```

Les objectifs

Deuxième paramètre : Utiliser une option choisie

l'option **- -recherche** pour effectuer une recherche d'un mot dans le dico



L'option est inconnue

Erreur option inconnue

l'option **- -pendu** pour jouer au pendu

Aide utilisateur

- Explications : Création de fonctions pour aider l'utilisateur : aide() et consulterAide()

```
aide
echo     "\n---------Aide du programme --------"
echo     '\nUtilisation : .\langstat.sh fichier [--option]'
echo "fichier : parametre obligatoire de type fichier"
echo     "Option :--recherche pour rechercher un mot"
echo     "Option :--pendu pour jouer au pendu "
echo     "\n---------Fin aide-----------------"
sleep 5



reponse ''
consulterAide
     -p "Voulez vous consulter l'aide ? 1-Oui 2-Non " reponse
        $reponse    '1'        $reponse    '2'

             -p  "Erreur : Tappez 1-Oui ou 2-Non" reponse

     $reponse    '1'
    aide
```

Trop de paramètres

- Si le premier paramètre est erroné, ou s'il y a trop de paramètres dans la ligne de commande

```
$#      2
    echo "Erreur : Trop de parametres ! "
    consulterAide
```

Vérification : validité du premier paramètre

```
fichier "$1"

    "compteur = 0 "

          "$fichier" -o      "$fichier" -o    "$fichier" -o      "$fichier" -o      "$fichier"


trouveDico    find -name dico.txt   wc -l

     $trouveDico      1        "$compteur"   "0"

echo "Un fichier dico.txt a été trouvé dans le repertoire courant: "
find -name dico.txt
    "compteur= compteur + 1 "


        "$fichier"
        echo "Erreur : le premier paramètre est vide"
            "$fichier"
        echo "Erreur :Le fichier n'existe pas "
        fichier ''
          "$fichier"
        echo "Erreur : Le fichier est un repertoire "
        fichier ''
            "$fichier" -o      "$fichier"
        echo "Erreur:Le fichier est vide ou inaccessible en lecture"
        fichier ''


     -p 'Merci d indiquer le chemin du fichier à traiter : ' fichier
```

Le premier paramètre est valide

Création du fichier statistiques, récupération de l'apparition d'une lettre dans un mot, tri et suppression du fichier

```
#Creation du fichier contenant les statistiques et du tableau contenant l alphabet
echo '' statLettres.txt

alphabet=( 'A' 'B' 'C' 'D' 'E' 'F' 'G' 'H' 'I' 'J' 'K' 'L' 'M' 'N' 'O' 'P' 'Q' 'R' 'S' 'T' 'U' 'V' 'W' 'X' 'Y' 'Z' )

#Parcours de chaque lettre composant le tableau
for letter in ${alphabet[@]}
do
#Recuperation du nombre de fois ou la lettre apparait
occurences=$( grep -i $letter $fichier | wc -l )

#On inscrit le resultat dans le fichier statLettres.txt >>
echo "$occurences - $letter" statLettres.txt
done


#Affichage du resultat et Tri du fichier statLettres.txt ; puis suppression du fichier
echo -e "\nSTATISTIQUES LETTRES DANS LE DICTIONNAIRE : "
sort -nr statLettres.txt
rm statLettres.txt
```

Option - -recherche

- Rappel du contexte :

L'utilisateur a renseigné le paramètre **- -recherche** en deuxième argument

Fonction recherche

```
#Fonction recherche
recherche(){
read -p 'Tappez le mot que vous rercherchez : ' mot
        #On verifie avec expression regulière si le mot recherché est trouvé dans le dictionnaire
        nbreMotTrouve=$( grep -Ei ^$mot$ $fichier | wc -l)

        #On verifie si il existe des mots commencant par le mot recherché
        nbreMotDebut=$( grep -Ei ^$mot $fichier | wc -l)
        if [ $nbreMotTrouve -ge 1 ];then
            echo "Mot trouvé  : "
            grep -Ein ^$mot$ $fichier

        #On ecrit dans le fichier motsDebut.txt les mots debutant par le mot recherché
        #On inscrit le numero de ligne (-n) et le mot trouvé
        #On affiche avec "less"
        elif [ $nbreMotDebut -ge 1 ] ;then
            echo '' motsDebut.txt
            echo "Il y a $nbreMotDebut  mot(s) débutant par $mot : " >> motsDebut.txt
            grep -Ein ^$mot $fichier >> motsDebut.txt
            less motsDebut.txt
            rm motsDebut.txt
        else
            echo -e "\nAucune correspondance a été trouvée "
        fi
```

Option --pendu

- Contexte :

le deuxième paramètre est **- -pendu** : On lance le script externe **pendu.sh**

```
"$2" = "--pendu" ]; then
 echo -e "\n-------------- JEU DU PENDU --------------"
./pendu.sh $fichier
 echo -e "\n-------------- FIN PARTIE ----------------"
```

```
reponsePendu="oui"

        while [ $reponsePendu = "oui" ]
        do

                jouerPendu $1
                read -p "Voulez vous rejouer au pendu : tappez (oui) ou autre touche pour quitter " reponsePendu
        done
```

- Dans le Fichier **pendu.sh** :

  On lance la fonction jouerPendu()

Fonction jouerPendu()

- Fonction jouerPendu():
- **Récupérer une ligne du dictionnaire et transformation en tableau**

```
jouerPendu ()
{
#variable pour compter le nombre de mots du dico.txt
nombreMots=$( wc -l $1 | cut -d' ' -f 1)
#nombre tiré au hasard
nbreRandom=$(( $RANDOM*$nombreMots/32767+1 ))

#option : servira d'option pour la commande sed
option=$nbreRandom
option="$option""p"
#grace à la commande sed on pourra obtenir le mot
#Exemple : sed -n 2p "dico.txt" #rend la 2 eme ligne
motAtrouver=$( sed -n $option $1 )

#compter le nombre de lettres et transformer le mot en " - - - - - - - - -"
nbreLettresMotatrouver=${#motAtrouver}
pendu=$( echo $motAtrouver | tr "[A-Z]" "-" )
#creation 2 tableau : array contiendra le mot à chercher et arrayPendu mot sous forme "- - -"
#identique :variable pour comparer le contenu des 2 tableaux
array=
arrayPendu=
identique="0"
#Transformer les strings en tableaux
for i in $( seq 0 $(( $nbreLettresMotatrouver-1 )) )
do
        j=$(( $i+1 ))
        array[$i]=$( echo $motAtrouver | cut -c $j-$j )
        arrayPendu[$i]=$( echo $pendu | cut -c $j-$j )
done
```

Fonction jouerPendu()

- Fonction comparer le mot à trouver et le pendu

```
comparerPenduMot
identique="1"
    k     seq 0 ${#array[@]}

    "${array[$k]}"  = "${arrayPendu[$k]}"
    identique="0"
```

- Afficher le mot sous forme « ----- » et indiquer le nombre de tentatives

```
echo  "\nMot à trouver de "    $nbreLettresMotatrouver - 1   " lettres : ${arrayPendu[@]}"

 "nbreChance = 9"
echo "Vous avez $nbreChance tentatives "
```

```
Mot à trouver de 13 lettres : - - - - - - - - - - - - -
Vous avez 9 tentatives
```

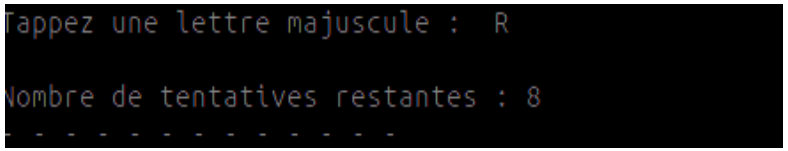
Fonction jouerPendu()

- La Partie peut débuter
- L'utilisateur entre la lettre :
  - vérifier la présence de la lettre
  - comparaison du pendu et mot à trouver
  - diminuer le nombres de tentatives
  - afficher le pendu

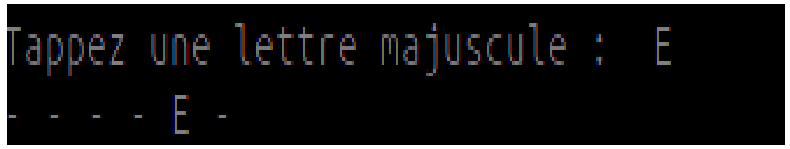
```
$nbreChance -gt 0 ]
do

#comparer les tableaux grace a la fonction pendu
comparerPenduMot

#La variable identique ="1" les 2Tableaux identiques et le joueur a gagne
if [ $identique = "1" ];then
    echo -e "\nBRAVO ! VOUS AVEZ GAGNE !!!"
exit
fi

#Le joueur est invite a entrer une lettre , on verifie la presence de la lettre dans le mot a trouver
read -p "Tappez une lettre majuscule :  " lettre
lettrePresence=`echo $motAtrouver | grep $lettre`
#teste de la valeur de retour $? de la commande precedente
if [ "$?" -eq 0 ];then
    #On recupere la valeur et on l'insere dans la case correspondante du pendu
    for k in `seq 0 ${#array[@]}`
    do
        if [ "${array[$k]}" = "$lettre" ];then
            arrayPendu[$k]=${array[$k]}
        fi

    done
#Lettre absente on dimunie le nombre de tentatives
else
    nbreChance=$(( nbreChance -1 ))
    echo -e "\nNombre de tentatives restantes : $nbreChance"

fi

#On affiche le pendu
echo ${arrayPendu[@]}

done
```

Quelques exemples de résultats obtenus

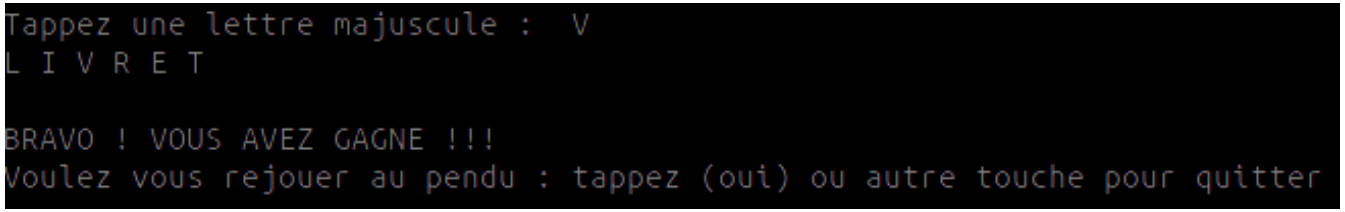
- Lettre absente



- Lettre présente



- **Le mot a été trouvé**



Le joueur a perdu

```
if [ $nbreChance -eq 0 ];then
echo -e "\nDOMMAGE ! VOUS AVEZ PERDU "
echo "Le mot à trouver était ${array[@]}"
fi
```

```
DOMMAGE ! VOUS AVEZ PERDU
Le mot à trouver était S P A T I A L I S I O N S
Voulez vous rejouer au pendu : tappez (oui) ou autre touche pour quitter
```

Merci de votre attention