```
#!/bin/bash
clear
#
#Test arguments
#
if [ $# -eq 0 ]; then
  echo No has introducido ningún argumento
  exit 1
fi

if [ $# -gt 1 ]; then
  echo Has introducido más de un argumento
  exit 1
fi

if [ "$1" != "conf.cfg" ]; then
  echo El argumento debe ser \"conf.cfg\"
  exit 1
fi

if !(test -f "conf.cfg"); then
  echo El fichero \"conf.cfg\" no está en el directorio o no existe,
creelo antes de ejecutar el script
  exit 1
fi
#
#Test
#
#Variables to know where are estadisticas.txt, diccionario.txt and their
paths
#
temp=(`head -1 conf.cfg|cut -d ":" -f 2`)
ubiDic=${temp%/*}
ficDic=${temp##*[/]}
temp=(`tail -1 conf.cfg|cut -d ":" -f 2`)
ubiEst=${temp%/*}
ficEst=${temp##*[/]}
ruta=(`pwd`)
if !(test -f $ubiEst/$ficEst); then
  touch estadisticas.txt
  ubiEst=(`pwd`)
  rm conf.cfg
  touch conf.cfg
  chmod 744 conf.cfg
  echo Diccionario: $ubiDic/$ficDic>>conf.cfg
  echo Estadisticas: $ubiEst/$ficEst>>conf.cfg
  echo Se ha creado el fichero estadisticas.txt
fi
#
#End of tests and begining of function menu
#
g=0
menu(){
clear
#
#Variables to know where are estadisticas.txt, diccionario.txt and their
paths every time we want to play
#
```

```
temp=(`head -1 conf.cfg|cut -c 13-70`)
ubiDic=${temp%/*}
ficDic=${temp##*[/]}
temp=(`tail -1 conf.cfg|cut -c 15-70`)
ubiEst=${temp%/*}
ficEst=${temp##*[/]}
ruta=(`pwd`)
echo "         Ahorcado"
echo "========================="
echo "J) JUGAR"
echo "E) ESTADISTICAS"
echo "C) CONFIGURACION"
echo "A) AYUDA"
echo "G) GRUPO"
echo "S) SALIR"
echo ""
echo "Juego del Ahorcado, introduzca opcion >>"

read opcion

case "$opcion" in
 j|J)
  clear
  game;;
 e|E)
  statistics
  cont;;
 c|C)
  config
  cont;;
 a|A)
  clear
  echo El juego consiste en adivinar una palabra tomada de forma
aleatoria del fichero \"diccionario.txt\"
  echo Para jugar pulsar j/J
  echo Para ver las estadísticas de juegos anteriores pulsar e/E
  echo Para cambiar la configuración del juego pulsar c/C, nos permitirá
cambiar la ubicación de los ficheros \"diccionario.txt\" y
\"estadisticas.txt\"
  echo Para ver los componentes del grupo pulsar g/G
  echo Para salir del menú de juego y abandonar el programa pulsar s/S
  echo Si pulsas cualquier otra opción que no está en el menú mostrará un
mensaje de error y volverá al menú
  echo ""
  cont;;
 g|G)
  echo Componentes del grupo:
  echo Juan Carlos Martín García
  echo Beatriz Botana Vázquez
  cont;;
 s|S)
  echo Has salido del menú
  exit 0;;
 *)
  echo Has introducido mal la opción, prueba de nuevo
  cont;;
esac
}
```

```
#
#End of menu function and begining of game function
#
game(){
 cd "$ubiDic"
 #
 #Variable that allows the two last numbers of the PID
 #
 pid=$((`ps -e|tail -1|cut -c 4-5`))
 #
 #Variable to know how many lines have Diccionario.txt
 #
 lines=$((`wc -l $ficDic|cut -d" " -f 6`))
 #
 #Random variable that we will use to know what word we will have to
guess
 #To make sure all of the words can be guessed for any Diccionario.txt
 #
 ranpid=$(( $(($pid+$RANDOM)) % $lines))
 let ranpid++
 #
 #Select the word we want to guess and convert to capital letters
 #
 word=(`sed -n $(($ranpid))p "$ficDic"`)
 WORD=(`echo $word|tr '[a-z]' '[A-Z]'`)
 cd "$ruta"
 #
 #Variables
 #
 time1=$SECONDS
 fails=0
 good=0
 #
 #Array with the letters of the word of Diccionario.txt
 #
 length=${#word}
 i=1
 while [ $i -le $length ];do
   letter=(`echo $word | cut -c $i|tr '[a-z]' '[A-Z]'`)
   wordletter[$i]=$letter
   let i++
 done
 echo ""
 #
 #Array with "-"
 #
 i=1
 while [ $i -le $length ];do
   symbol[$i]="-"
   let i++
 done
 #
 #GAME
 #
 j=0
 tries=0
 while [ $fails -ne 8 ];do
   #
```

```
#Show the wrong letters wrote by the user
#
fai=$fails
if [ $fai -ne 0 ];then
  printf "Letras falladas: "
  while [ $fai -ne 0 ];do
   printf ${wrong[$fai]}" "
   let fai--
   if [ $fai -eq 0 ];then
     printf "       Estás "$ahorcado
   fi
  done
fi
echo ""
echo ""
#
#Show the symbols
#
i=0
while [ $i -ne $length ];do
  let i++
  for symb in "${symbol[$i]}";do
    printf "$symb"
  done
done
echo ""
#
#Ask for a letter and convert to capital letter
#
let j++
let tries++
echo "Introduce una letra, intento:" $tries
read X[$j]
t=(`echo ${X[$j]} | tr '[a-z]' '[A-Z]'`)
x[$j]=$t
clear
rt=0
if [ "${X[$j]}" = ñ ];then
   x[$j]=Ñ
   t=Ñ
   rt=1
fi
if ! [ "${t}" = "$(echo ${t}|egrep '^[A-Z]*$')" ];then
   if [ "${t}" = "$(echo ${t}|egrep '^[0-9]*$')" ];then
     let tries--
     let j--
   echo "No se permiten números"
   continue
   elif [ "${x[$j]}" = Ñ ];then
   printf ""
   else
     let tries--
     let j--
   echo "El caracter: "$t" no es válido"
   continue
   fi
fi
#
```

```
   #Test if the user has wrote it before
   #
   b=0
   k=$(($j-1))
     while [ $k -ne 0 ];do
        last=${x[$j]}
        if [ "${x[$j]}" = "" ];then
          let k--
        else
          if [ "${x[$k]}" = $last ];then
           let j--
           let tries--
           echo "Ya has introducido: "$t" antes"
           echo "No se contará como intento ni como fallo"
           echo ""
           b=1
           break
          else
          let k--
          fi
        fi
   done
   if [ $b -eq 1 ];then
      continue
   fi
   #
   #Test if the user is trying to write more than one letter
   #
   if test "${#x[$j]}" = 1 -o "$rt" = 1 ; then
      #
      #Test if the letter that user writes is one of the word he has to
guess
      #
      k=1
      p=0
        while [ "$k" -le $length ];do
          if [ "${wordletter[$k]}" = ${x[$j]} ];then
          symbol[$k]=${x[$j]}
          let k++
          p=$k
          let good++
        fi
        let k++
      done

      if [ $p -eq 0 ];then
      fail
      fi
      if [ $good -ge $length ];then
      break;
      fi
   elif [ ${#t} -eq $length ];then
      if [ "$t" = $WORD ];then
        good=$length
        break;
      else
      fail
      fi
```

```
   else
     echo "Has introducido "${#X[$j]}" caracteres, introduce solo uno o
la palabra entera"
     let j--
     let tries--
   fi
 done
 time2=$SECONDS
 #
 #If the user fails 10 times the game will say him that he lose and if
not that he wins, later write the information in Estadisticas.txt
 #
 if [ $good -eq $length ];then
   echo $word
   echo "¡Enhorabuena! lo has conseguido en: "$tries" intentos"
 elif [ $fails -eq 8 ];then
   echo "Has fallado 8 veces, has perdido."
   echo "La palabra a adivinar era: "$word
 fi
 #
 #End of game
 #
 time=$(($time2-$time1))
 cd "$ubiEst/"
 echo $ranpid" "$pid" "$word" "$tries" "$fails" "$time>>"$ficEst"
 cd "$ruta"
 g=1
 cont
}
#
#End of function game and begining of statistics function
#
statistics(){
 clear
 #Variable to know where is statistics file
 cd "$ubiEst"
 totalPlay=(`wc -l "$ficEst"`)
 echo TOTALES
 echo "   · Número total de partidas jugadas: "$totalPlay
 echo MEDIAS
 #
 #To know the media of tries, fails and time
 #
 for x in 4 5 6; do
  count=(`wc -l "$ficEst"`)
  totalTries=0
  i=1
  while [ $i -le $count ]; do
    m=(`cut -d " " -f $x "$ficEst"|sed -n "$i"p`)
    totalTries=$(($m+$totalTries))
    let i++
  done
  if [ "$x" = 4 ];then
    y="Número medio de intentos"
  elif [ "$x" = 5 ];then
    y="Número medio de fallos"
  elif [ "$x" = 6 ];then
    y="Media de tiempo de los intentos"
```

```
  fi
  printf "  · $y: "
  echo "($totalTries / $count)" | bc -l;
 done
  echo MEJORES JUGADAS
 for i in 6 5 4; do
   if [ "$i" = 6 ];then
     y="Tiempo de la jugada más corta"
   elif [ "$i" = 5  ];then
     y="Número mínimo de fallos"
   elif [ "$i" = 4 ];then
     y="Número mínimo de intentos"
   fi
   best=(`sort -t" " -k"$i"n "$ficEst" | head -1|cut -d " " -f $i`)
   if [ "$i" != 6 ];then
     echo "  · $y: $best"
   else
     min=$(($best/60))
     seg=$(($best%60))
     echo "  · $y: "$min"min "$seg"s"
   fi
 done
 echo PEORES JUGADAS
 for i in 6 5 4; do
  if [ "$i" = 6 ];then
     y="Tiempo de la jugada más larga"
   elif [ "$i" = 5  ];then
     y="Número máximo de fallos"
   elif [ "$i" = 4 ];then
     y="Número máximo de intentos"
   fi
  worst=(`sort -t " " -k"$i"n "$ficEst" | tail -1 |cut -d " " -f $i`)
  if [ "$i" != 6 ];then
      echo "  · $y: $worst"
   else
     min=$(($worst/60))
     seg=$(($worst%60))
     echo "  · $y: "$min"min "$seg"s"
   fi
 done
 cd "$ruta"
}
#
#End of function statistics and begiging of config function
#
config(){
 #
 #Ask if the user wants to change the configuration
 #
 echo Vas a cambiar la configuración, ¿estás seguro de que quieres
continuar? \(s/n\)
 read conf
 #
 #Test the answer
 #
 case "$conf" in
  s|S)
```

```
  echo Introduce la ruta donde quieres que se almacene Diccionario.txt
\(excluyendo el fichero .txt\)
  read newubiDic
    if test -d "$newubiDic";then
       if [ "$ubiDic/" != $newubiDic ];then
          case "$newubiDic" in
            */)
              cd "$ubiDic"
              mv $ficDic "$newubiDic"
              echo Fichero diccionario.txt movido correctamente
              cd "$ruta"
              rm conf.cfg
              touch conf.cfg
              chmod 744 conf.cfg
              echo "Diccionario: "$newubiDic$ficDic>>conf.cfg
              ;;
            *)
              cd "$ubiDic"
              mv $ficDic "$newubiDic/"
              echo Fichero diccionario.txt movido correctamente
              cd "$ruta"
              rm conf.cfg
              touch conf.cfg
              chmod 744 conf.cfg
              echo "Diccionario: "$newubiDic/$ficDic>>conf.cfg
              ;;
          esac
       else
       rm conf.cfg
       touch conf.cfg
       chmod 744 conf.cfg
       echo "Diccionario: "$ubiDic"/"$ficDic>>conf.cfg
       fi
    else
      echo $newubiDic " no existe o no es un directorio "
      rm conf.cfg
      touch conf.cfg
      chmod 744 conf.cfg
      echo "Diccionario: "$ubiDic"/"$ficDic>>conf.cfg
    fi
  echo Introduce la ruta donde quieres que se almacene Estadisticas.txt
\(excluyendo el fichero .txt\)
  read newubiEst
    if test -d "$newubiEst";then
      if [ "$ubiEst/" != $newubiEst ];then
         case "$newubiEst" in
            */)
            cd "$ubiEst"
              mv $ficEst "$newubiEst"
                  echo Fichero estadisticas.txt movido correctamente
            cd "$ruta"
            echo "Estadisticas: "$newubiEst$ficEst>>conf.cfg
            ;;
            *)
            cd "$ubiEst"
            mv $ficEst "$newubiEst/"
                  echo Fichero estadisticas.txt movido correctamente
            cd "$ruta"
```

```
            echo "Estadisticas: "$newubiEst"/"$ficEst>>conf.cfg
            ;;
          esac
       else
         echo "Estadisticas: "$ubiEst"/"$ficEst>>conf.cfg
       fi
     else
          echo $newubiEst " no existe o no es un directorio"
          echo "Estadisticas: "$ubiEst"/"$ficEst >>conf.cfg
     fi
     cont;;
  *)
   echo No se cambiará la configuración
   cont;;
 esac
}
#
#End of config function and begining of fail function
#
fail(){
        let fails++
        wrong[$fails]=${x[$j]}
      case $fails in
        1) ahorcado="A";;
        2) ahorcado="AH";;
        3) ahorcado="AHO";;
        4) ahorcado="AHOR";;
        5) ahorcado="AHORC";;
        6) ahorcado="AHORCA";;
        7) ahorcado="AHORCAD";;
        8) ahorcado="AHORCADO";;
      esac
}
#
#End of fail function and begining of askExit function
#
cont(){
   echo ""
   echo \"Pulsa una tecla para continuar\"
   read;
     menu
}
#
#End of cont function
#
menu
```