#99 Unscramble: A Word Game

If you've seen the Jumble game in your newspaper or played word games at all, you're familiar with the basic concept of this game: A word is picked at random and then scrambled. Your task is to figure out and guess what the original word is in the minimum number of turns.

The Code

```
#!/bin/sh

# unscramble - Picks a word, scrambles it, and asks the user to guess
#    what the original word (or phrase) was.

wordlib="/usr/lib/games/long-words.txt"
randomquote="$HOME/bin/randomquote"    # Script #76

scrambleword()
{
  # Pick a word randomly from the wordlib, and scramble it.
  # Original word is $match and scrambled word is $scrambled

  match="$($randomquote $wordlib)"

  echo "Picked out a word!"

  len=$(echo $match | wc -c | sed 's/[^[:digit:]]//g')
  scrambled=""; lastval=1

  for (( val=1; $val < $len ; ))
  do
    if [ $(perl -e "print int rand(2)") -eq 1 ] ; then
      scrambled=$scrambled$(echo $match | cut -c$val)
    else
      scrambled=$(echo $match | cut -c$val)$scrambled
    fi
    val=$(( $val + 1 ))
  done
}

if [ ! -r $wordlib ] ; then
  echo "$0: Missing word library $wordlib" >&2
  echo "(online: http://www.intuitive.com/wicked/examples/long-words.txt" >&2
  echo "save the file as $wordlib and you're ready to play!)" >&2
  exit 1
fi

newgame=""; guesses=0; correct=0; total=0

until [ "$guess" = "quit" ] ; do

  scrambleword

  echo ""
  echo "You need to unscramble: $scrambled"
  guess="??" ; guesses=0
  total=$(( $total + 1 ))

  while [ "$guess" != "$match" -a "$guess" != "quit" -a "$guess" != "next" ]
  do
    echo ""
    echo -n "Your guess (quit|next) : "
    read guess

    if [ "$guess" = "$match" ] ; then
      guesses=$(( $guesses + 1 ))
      echo ""
      echo "*** You got it with tries = ${guesses}!  Well done!! ***"
      echo ""
      correct=$(( $correct + 1 ))
    elif [ "$guess" = "next" -o "$guess" = "quit" ] ; then
      echo "The unscrambled word was \"$match\". Your tries: $guesses"
    else
      echo "Nope. That's not the unscrambled word. Try again."
      guesses=$(( $guesses + 1 ))
    fi
  done
done

echo "Done. You correctly figured out $correct out of $total scrambled words."

exit 0
```

How It Works

To randomly pick a single line from a file, this script uses Script #76, *Displaying Random Text*, even though it was originally written to work with web pages. Like many good Unix utilities, it turns out to be a useful building block in other contexts than the one it was intended for:

```
match="$($randomquote $wordlib)"
```

The toughest part of this script was figuring out how to scramble a word. There's no handy Unix utility for that, but fortunately it turns out that if we assemble the scrambled word by going letter by letter through the correctly spelled word and randomly adding each subsequent letter to the scrambled sequence at either the beginning or the end of the sequence, we quite effectively scramble the word differently and unpredictably each time:

```
if [ $(perl -e "print int rand(2)") -eq 1 ] ; then
  scrambled=$scrambled$(echo $match | cut -c$val)
else
```

```
  scrambled=$(echo $match | cut -c$val)$scrambled
fi
```

Notice where $scrambled is located in the two lines: In the first line the added letter is appended, while in the second it is prepended.

Otherwise the main game logic should be easily understood: The outer while loop runs until the user enters quit as a guess, while the inner loop runs until the user either guesses the word or types next to skip to the next word.

Running the Script

This script has no arguments or parameters, so just type in the name, and you're ready to play!

The Results

```
$ unscramble
Picked out a word!

You need to unscramble: ninrenoccg

Your guess (quit|next) : concerning

*** You got it with tries = 1!  Well done!! ***

Picked out a word!

You need to unscramble: esivrmipod

Your guess (quit|next) : quit
The unscrambled word was "improvised". Your tries: 0
Done. You correctly figured out 1 out of 2 scrambled words.
```

Clearly an inspired guess on that first one!

Hacking the Script

Perhaps some method of offering a clue would make this game more interesting or, alternatively, a flag that requests the minimum word length that is acceptable. To accomplish the former, perhaps the first *n* letters of the unscrambled word could be shown for a certain penalty in the scoring; each clue requested would show one additional letter. For the latter, you'd need to have an expanded word dictionary, as the one included with the script has a minimum word length of ten letters, which makes it rather tricky!