TP – Contrôle langage C – N°1
Ciel 1 - 2023-2024

Jeu du pendu

Principe :

Un mot, connu uniquement par le système, est choisi aléatoirement. Le système affiche une série de tirets correspondant au nombre de lettres dans le mot à deviner.

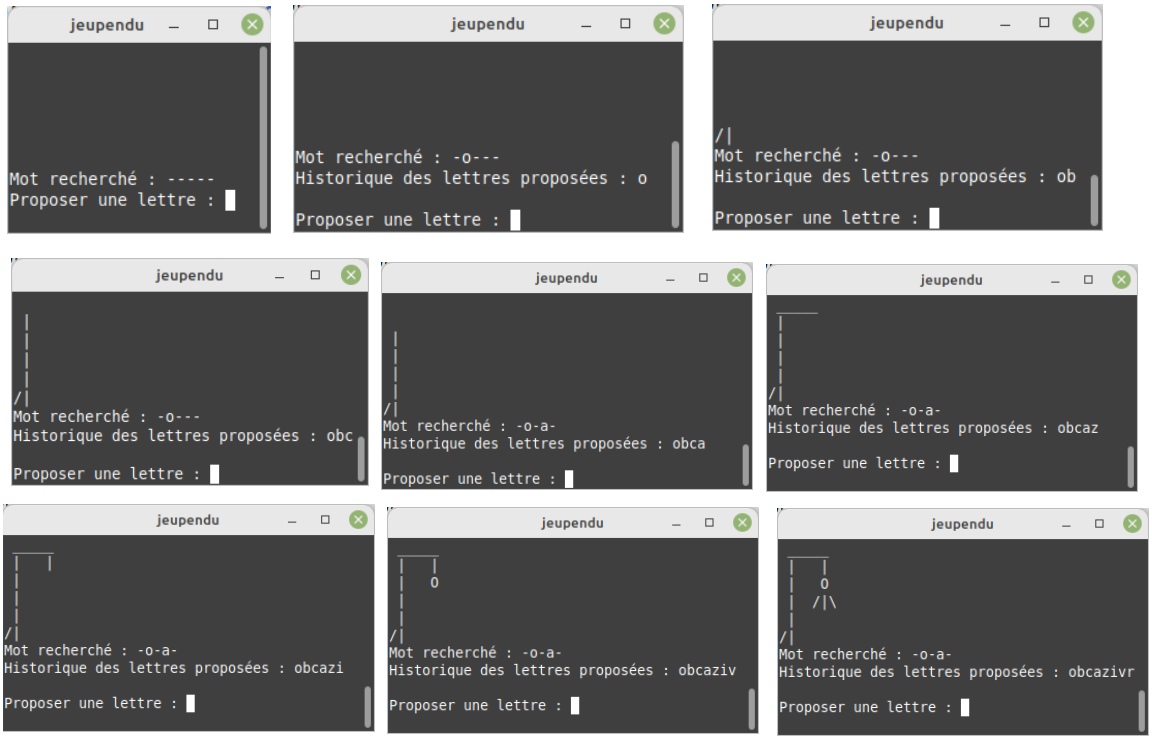
Le joueur propose une lettre. À chaque proposition valide, le système révèle les emplacements de cette lettre dans le mot en remplaçant les tirets par la lettre découverte.

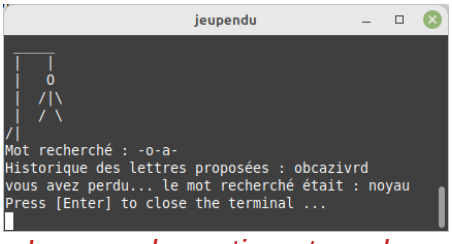
Le joueur peut proposer autant de lettres que nécessaire pour deviner le mot. Cependant, après 7 propositions incorrectes, le joueur est 'pendu' et la partie est perdue.

Si le joueur découvre le mot, le système affiche le nombre de tentatives qui ont été nécessaires pour y parvenir.

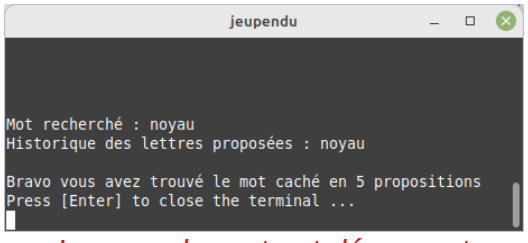
Exemple :

Le mot recherché ici est « **noyau** » :





Lorsque la partie est perdue



Lorsque le mot est découvert

Travail à réaliser :

1. Réalisez un nouveau projet du type application en **langage C** sous **NetBeans** portant le nom **JeuDuPendu**. À la fin de la séance vous déposerez votre travail dans le dossier à votre nom dans le dossier :
   /home/USERS/ELEVES/CIEL2023/TpCtrl/ TPCtrl1_C.
2. Créez dans votre projet les fichiers **gestionPendu.h** et **gestionPendu.c** qui constituerons la bibliothèque de fonctions nécessaires à votre programme.

   Ajoutez les lignes de code suivantes au fichier d'en-tête de votre bibliothèque :

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/random.h>
```

Définissez également les constantes suivantes :

`NB_MOTS` avec la valeur de 5

`NB_MAX_LETTRES_PROPOSEES` et **`TAILLE_MAX_MOTS`** avec les valeurs 50

On donne la fonction suivante :

```
const char *ObtenirMotAleatoire()
{
    static const char *listeMots[NB_MOTS] = {"bonjour", "maison", "noyau", "temps", "blond"};
    unsigned int valeurAleatoire;
    getrandom(&valeurAleatoire, sizeof(valeurAleatoire), 0);
    return listeMots[valeurAleatoire % NB_MOTS];
}
```

3. Codez le prototype de cette fonction dans le fichier **gestionPendu.h** et recopiez sa définition dans le fichier **gestionPendu.c**.
4. Écrivez le code de la fonction **`InitialiserMot`**, prenant en paramètre d'entrée/sortie un tableau de caractères nommé **`_motCourant`** et en paramètre d'entrée **`_nbLettres`**, un entier représentant le nombre de lettres dans le mot à découvrir. Cette fonction permet de placer un tiret à la place de chaque lettre dans la variable **`_motCourant.`**`Elle` ne retourne aucune valeur. Ajoutez le prototype de cette fonction au fichier d'en-tête de votre bibliothèque de fonctions.

On donne le prototype de la fonction suivante :

```
int PlacerLettre(const char _motATrouver[], char _motCourant[],const char _lettreProposee) ;
```

5. Implémentez la fonction **`PlacerLettre`** dont le rôle est de remplacer les tirets dans la chaîne **`_motCourant`** par la lettre proposée en fonction de sa position dans le mot à deviner. Cette fonction doit retourner le nombre de fois où la lettre a été placée dans le mot.
6. Après l'inclusion des librairies **`string.h`** et **`stdbool.h`** dans le fichier d'en-tête de votre bibliothèque de fonctions, implémentez la fonction dont le prototype est donné ci-après. Elle retourne la valeur **`true`** si les deux chaînes sont identiques, **`false`** sinon.

```
bool VerifierProposition(const char _motATrouver[],const char _motCourant[]) ;
```

Vous utiliserez pour cela la fonction **`strcmp`** de la librairie **string.h** (voir annexes).

7. Complétez le code de la fonction **`AfficherPendu`**. Cette fonction prend en paramètre le nombre de tentatives infructueuses et n'a pas de valeur de retour.

```
void AfficherPendu(const int _nbErreur)
{
    char *pendu[8][6] = {
        "       \n", // pas d'erreur
        "       \n",
        "       \n",
        "       \n",
        "       \n",
        "       \n",

        "       \n", // 1 erreur
        "       \n",
        "       \n",
        "       \n",
        "       \n",
        "/|     \n",

        "       \n", // 2 erreurs
        " |     \n",
        " |     \n",
        " |     \n",
        " |     \n",
        "/|     \n",

        " ____\n", // 3 erreurs
        " |     \n",
        " |     \n",
        " |     \n",
        " |     \n",
        "/|     \n",

          // tableau à compléter

        " _____\n", // 6 erreurs
        " |   |  \n",
        " |   O  \n",
        " |  /|\\ \n",
        " |      \n",
        "/|      \n",

        " _____ \n", // 7 erreurs
        " |   |  \n",
        " |   O  \n",
        " |  /|\\ \n",
        " |  / \\ \n",
        "/|      \n",
    };

   // code à ajouter ici pour l'affichage du pendu en fonction du nombre de tentatives infructueuses
}
```

8. On donne maintenant un extrait du fichier **main.c**. Complétez le code pour réaliser les traitements attendus.

```
int main(int argc, char** argv)
{
    char motRecherche[TAILLE_MAX_MOTS] ;                    // mot à deviner
    char motCourant[TAILLE_MAX_MOTS] ;                      // mot au cours du jeu
    char historique[NB_MAX_LETTRES_PROPOSEES] = {0} ;  // lettres déjà proposées
    char lettre;
    int  cptPropositions = 0 ;      // compteur de proposition effectuée
    int  nbErreur = 0 ;             // nombre de tentative infructueuses
    strncpy(motRecherche,ObtenirMotAleatoire(),TAILLE_MAX_MOTS-1);
   // Appel de la fonction pour initialiser le motCourant
   // Appel de la fonction pour afficher le pendu
    printf("Mot recherché : %s\n", motCourant);
    printf("Proposer une lettre : ");
    do
    {
        scanf(" %c", &lettre);           // Saisie de la lettre en évitant les '\n' intempestifs.
        printf("\033[2J\033[1;1H");   // Pour effacer la console
        historique[cptPropositions++] = lettre;
        // si la lettre n'a pas pu être placé dans le mot courant, le nombre d'erreurs est incrémenté
        // Appel de la fonction pour afficher le pendu
        printf("Mot recherché : %s\n", motCourant);
        printf("Historique des lettres proposées : %s\n", historique);
        // si on n'est pas à la fin du jeu on affiche "Proposer une lettre"
    } while (/* Ce n'est pas la fin de la partie */);

    if (nbErreur == 7)
        printf("vous avez perdu... le mot recherché était : %s\n",motRecherche);
    else
      // Afficher le message attendu en cas de victoire

    return (EXIT_SUCCESS);
}
```

9. Modifiez le programme principale pour que l'utilisateur puisse jouer autant de fois qu'il le souhaite sans relancer le programme.

Annexes :

`Fonction strcmp`

```
int strcmp(const char *chaine1, const char *chaine2);
```

Cette fonction prend deux chaînes de caractères en entrée (**`chaine1`** et **`chaine2`**) et renvoie un entier :

- Si les deux chaînes sont identiques, strcmp retourne 0.
- Si chaine1 est lexicographiquement inférieure à chaine2, la fonction retourne un entier négatif.
- Si chaine1 est lexicographiquement supérieure à chaine2, la fonction retourne un entier positif.

`Fonction strncpy`

```
char *strncpy(char *destination, const char *source, size_t n);
```

Elle est utilisée pour copier une chaîne de caractères dans une autre, mais avec une limite sur le nombre de caractères à copier. Les paramètres de **`strncpy`** sont :

- **`destination`** : le pointeur vers la chaîne de caractères où la copie sera effectuée.
- **`source`** : le pointeur vers la chaîne de caractères à copier.
- **`n`** : le nombre maximal de caractères à copier depuis la chaîne source vers la destination.