Journal : sur le fonctionnement du jeu Wordle

Posté par **steph1978** le 13/01/22 à 11:39. **Licence CC By-SA.**
Étiquettes : **rétro-ingénierie**, **jeu**, **puzzle**, **javascript**

J'ai croisé **le jeu wordle** dans plusieurs articles récents lors de **ma veille**.
C'est un peu comme un **mastermind**[W] mais avec des mots du dictionnaire. Il y a un mot par jour à deviner, de cinq lettres. Le mot d'hier était 'favor'.
Vous proposez un mot du dictionnaire (lequel ?), le jeu vous dit quelles lettres sont dans le mot à trouver et quelles lettres sont à la bonne place, quelles lettres n'y sont pas.
J'ai joué mais je ne suis pas doué. Je connais combien de mots de cinq lettres en anglais …

Par contre mon pc connaît plus de 15k mots d'anglais de cinq lettres, oui oui. Et il est capable de valider ces mots par une liste de règles (contient un "a", a un "e" à telle position, ne contient pas de "p", …). J'ai pu trouver 'favor' en quatre essais à coup de `grep` et `grep -v`. C'est donc un travail pour un ordinateur.

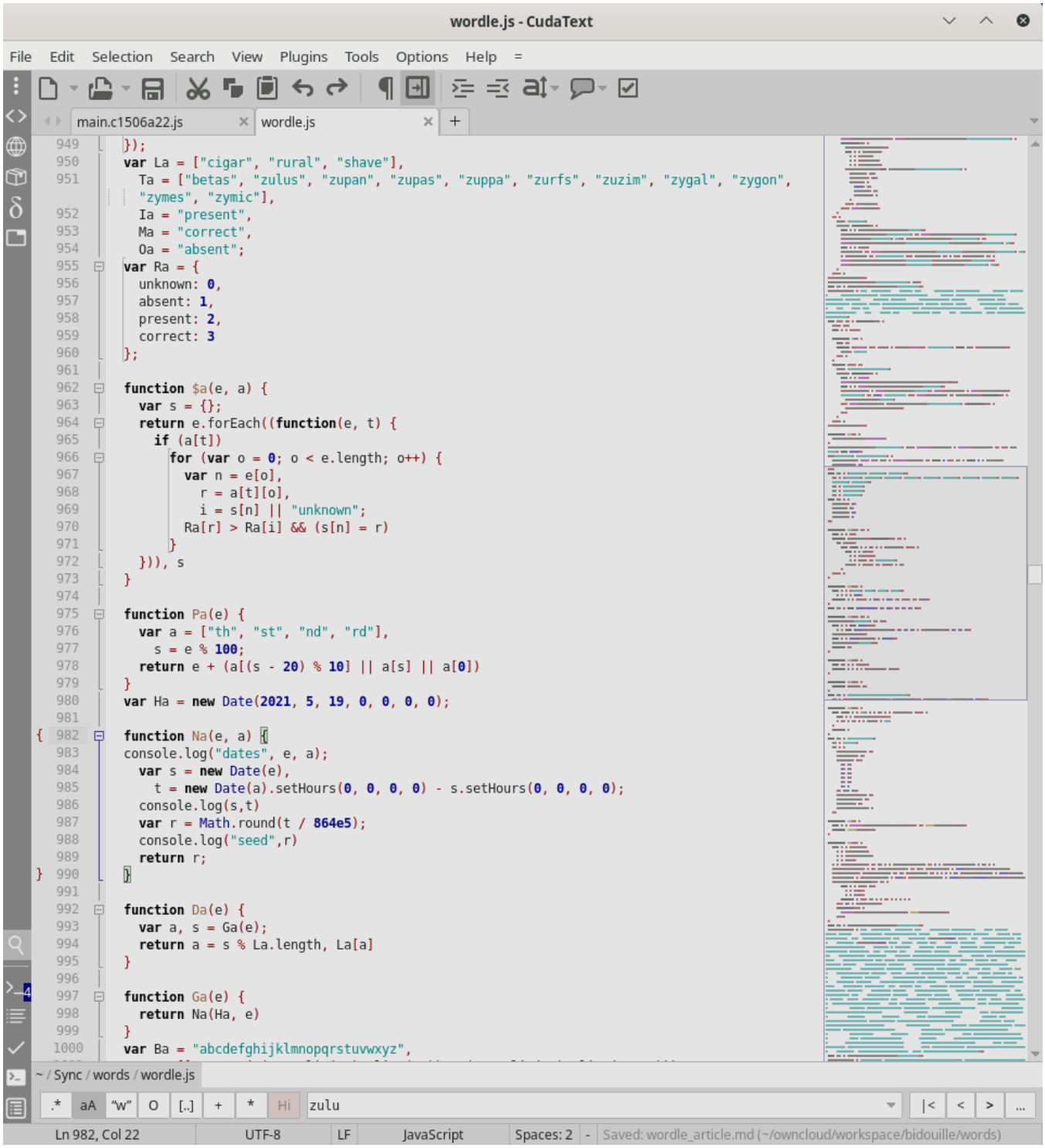
Je cherche tout d'abord à comprendre comment marche le jeu. Je lance le lien avec les outils de dev Firefox ouverts (`F12`). Je m'attends à des aller-retours avec le serveur pour valider les propositions de mots. Mais rien, tout est côté client ! Et quand c'est côté client, on regarde le source (`CTRL+U`).

Il y a un gros script de 177kB, très obfusqué, clairement passé dans la moulinette d'un builder JS. Je l'ouvre dans **cudatext**. Je browse un peu et vois rapidement une liste de mots de cinq lettres (le dictionnaire) … et juste après une autre liste de mots de cinq lettre (?).

J'extrais les deux listes et compte les mots : 2315 dans la première, 10657 dans la seconde, pas loin de mes 15k. À leur tête, je comprends que la première liste sont des mots courants, qui seront proposé à la devinette et que la deuxième est une liste de mots autorisés car dans le dictionnaire anglais mais beaucoup moins courant donc moins drôle à jouer.

En résumé, le jeu prends chaque jour un mot dans la première liste et valide les propositions du joueur sur l'union des deux listes.

Je suis assez curieux de savoir comment est choisi le mot du jour. Dans le javascript, je supprime la quasi totalité les mots des listes. Reste 77kB quand même. Je cherche **format javascript** sur le web et tombe sur **un outil** qui accepte de me reformater le script. Je remets ça dans cudatext, cherche un mot que j'ai laissé.



Je vois que la liste (`La`) est utilisée dans une fonction (`Na`) un peu plus bas. J'instrumente à coup de `console.log` et je remets ça dans la source HTML. Miracle ou presque le script reformaté fonctionne parfaitement et mes traces apparaissent. L'algo est des plus simple : date du jour - date de début du jeu, en jours => index du mot dans la première liste. La liste n'étant pas triée, cela donne un mot "aléatoire".

Pour preuve, le mot d'aujourd'hui :

Promis je ne vous spoile pas le reste.

Exercice laissé au lecteur ou à moi même : implémenter le jeu en client-serveur afin de ne pas divulguer la liste et pourvoir tenir un leaderborad (et blinder le truc de pubs et de cryptominers pour faire des bouzoufs).

SUTOM

Posté par **Benoît Sibaud** (**site Web personnel**) le 13/01/22 à 13:23. Évalué à 7 (+4/-0).

> On voit passer aussi du SUTOM **https://sutom.nocle.fr/** (framagit **https://framagit.org/JonathanMM/sutom**), en plus de Wordle. Par exemple les derniers gazouillis de B. Tréguier sont Wordle **https://twitter.censors.us/btreguier/status/1481565919402987521** et SUTOM **https://twitter.censors.us/btreguier/status/1481522325380120578**

Re: SUTOM

Posté par **Frédéric COIFFIER** le 13/01/22 à 13:59. Évalué à 4 (+3/-0).

> SUTOM pour ne pas dire Motus ?

Re: SUTOM

Posté par **Gil Cot** le 13/01/22 à 17:50. Évalué à 3 (+1/-0).

> Le verlan certainement pour éviter les poursuites de **Fr2 qui lui-même n'a pas repris Lingo pour les mêmes raisons**[W]. Comme qui dirait, c'est **quasiment comme l'un et presque comme le notre** haha.
>
> --
>
> "It is seldom that liberty of any kind is lost all at once." — David Hume

Re: SUTOM

Posté par **steph1978** le 13/01/22 à 16:06. Évalué à 4 (+2/-0).

> Merci pour le lien!
>
> SUTOM #6 3/6
>
> ▩▩▤▤◎◎
> ▩▩▤◎◎▤
> ▩▩▩▩▩▩
>
> **https://sutom.nocle.fr**

Autre approche : grep

Posté par **chimrod** (**site Web personnel**) le 13/01/22 à 14:26. Évalué à 9 (+7/-0). Dernière modification le 13/01/22 à 14:29.

> Note : je me base sur la liste des mots présents dans le dictionnaire américain de Debian, accessible via le paquet `wamerican` et listé dans le fichier `/usr/share/dict/american-english`
>
> Il est également possible de choisir ses mots en fonction de la **quantité d'information**[W] contenus dans chacun d'eux. Par exemple, le dictionnaire anglais contient 102.774 entrées :
>
> ```
> $ wc -l american-english
> 102774 american-english
> ```
>
> et l'on se rend compte que le `a` est présent dans 52% d'entre eux :
>
> ```
> $ grep -i "a" american-english  | wc -l
> 53715
> ```
>
> Si l'on choisi un mot avec un `a`, sans meme savoir quel sera le résultat, nous éliminons la moitié des mots possibles. Comme nous avons 5 lettres possibles, autant choisir celles qui permettent de discréminer un max de résultat. Et c'est parti pour une plongée en bash pour trouver la solution !
>
> ```
> $ grep -o . <<< "abcdefghijklmnopqrstuvwxyz"| while read letter; do echo $letter $(grep -i $letter american-
> a 53715
> …
> e 65692
> …
> n 47826
> …
> r 50018
> …
> t 44232
>
> $ grep "^.....$"  american-english | grep -i "a" | grep -i "r" | grep -i "n" | grep -i "t"
> Brant
> Grant
> Rutan
> Trina
> grant
> ```

```
rants
train
```

essayons `train` avec le mot du jour : le `a` est reconnu, mais pas les autres lettres ! Parfait, il ne reste que 878 mots présents :

```
$ grep "^.....$"  american-english | grep -i "a" | grep -iv "[trin]" > /tmp/iter1
$ wc -l /tmp/iter1
878
```

On recommence notre itération :

```
$ grep -o . <<< "abcdefghijklmnopqrstuvwxyz"| while read letter; do echo $letter $(grep -i $letter /tmp/iter
b 124
c 169
d 181
e 370
h 135
l 342
m 175
o 137
p 139
s 504
y 139

$ grep "^.....$"  american-english | grep -i "s" | grep -i "l" | grep -i "e" | grep -i "d" | grep -i "m"
melds
```

31 mots restants ! On s'approche !

```
$ grep "^.....$"  /tmp/iter1 | grep -i "a" | grep -i "e" | grep -iv "[trinmlds]" > /tmp/iter2
$ wc -l /tmp/iter2
31 /tmp/iter2
$ grep -o . <<< "abcdefghijklmnopqrstuvwxyz"| while read letter; do echo $letter $(grep -i $letter /tmp/iter
b 6
c 11
f 2
g 8
h 9
k 5
o 2
p 9
q 2
u 8
v 6
w 3
x 1
y 4
z 1

$ grep "^.....$"  /tmp/iter2 | grep -i "c" | grep -i "h" | grep -i "p"
```

```
cheap
peach
```

Prenons-en un des deux, et faisant notre test : plus que 14 possibilité, et il reste 3 coups !

```
$ grep "^.....$"  /tmp/iter1 | grep -i "a" | grep -i "e" | grep -iv "[trinmldspch]" > /tmp/iter3
$ wc -l /tmp/iter3
14 /tmp/iter3

$ grep -o . <<< "abcdefghijklmnopqrstuvwxyz"| while read letter; do echo $letter $(grep -i $letter /tmp/iter
b 4
g 5
k 4
o 2
q 2
u 6
v 4
w 3

$ grep "^.....$"  /tmp/iter3 | grep -i "u"
…
quake
```

Plus que 5 mots dans la liste, et encore deux coup restants ! Je pense qu'on est bon !

```
$ grep "^.....$"  /tmp/iter3 | grep -i "a" | grep -i "e" | grep -iv "[trinmldspchquk]"
abbey
above
agave
gaffe
weave
```

En fait, si l'on tient compte des informations sur le placement des lettres, on sait à ce moment là que le `e` n'est pas en dernière position, et il ne reste plus qu'une seule possibilité.

Y a-t-il un volontaire pour mettre tout ça dans un petit scritp interractif qui s'occupe de nous trouver la solution ? :)

Re: Autre approche : grep

Posté par **chimrod** (**site Web personnel**) le 13/01/22 à 14:42. Évalué à 3 (+1/-0). Dernière modification le 13/01/22 à 14:43.

Notes en vrac :

- ici, je me suis contenté d'appliquer un algorithme de type **arbre de décision**[W], afin de choisir quel est l'élément le plus pertinent à retenir.
- avec le français, cela aurait été plus compliqué, à cause des accents dans les mots qu'il faut transformer pour uniformiser les lettres
- enfin, j'ai commencé l'analyse en partant du dictionnaire complet, cela aurait été plus efficace si j'avais dès le début choisi les mots de 5 lettres pour identifier les fréquences.

Re: Autre approche : grep

Posté par **Gil Cot** le 13/01/22 à 18:36. Évalué à 6 (+4/-0).

Je remplacerais bien :

```
grep -i "a" american-english  | wc -l
```

par (pas juste plus court mais un processus en moins aussi) :

```
grep -ic a american-english
```

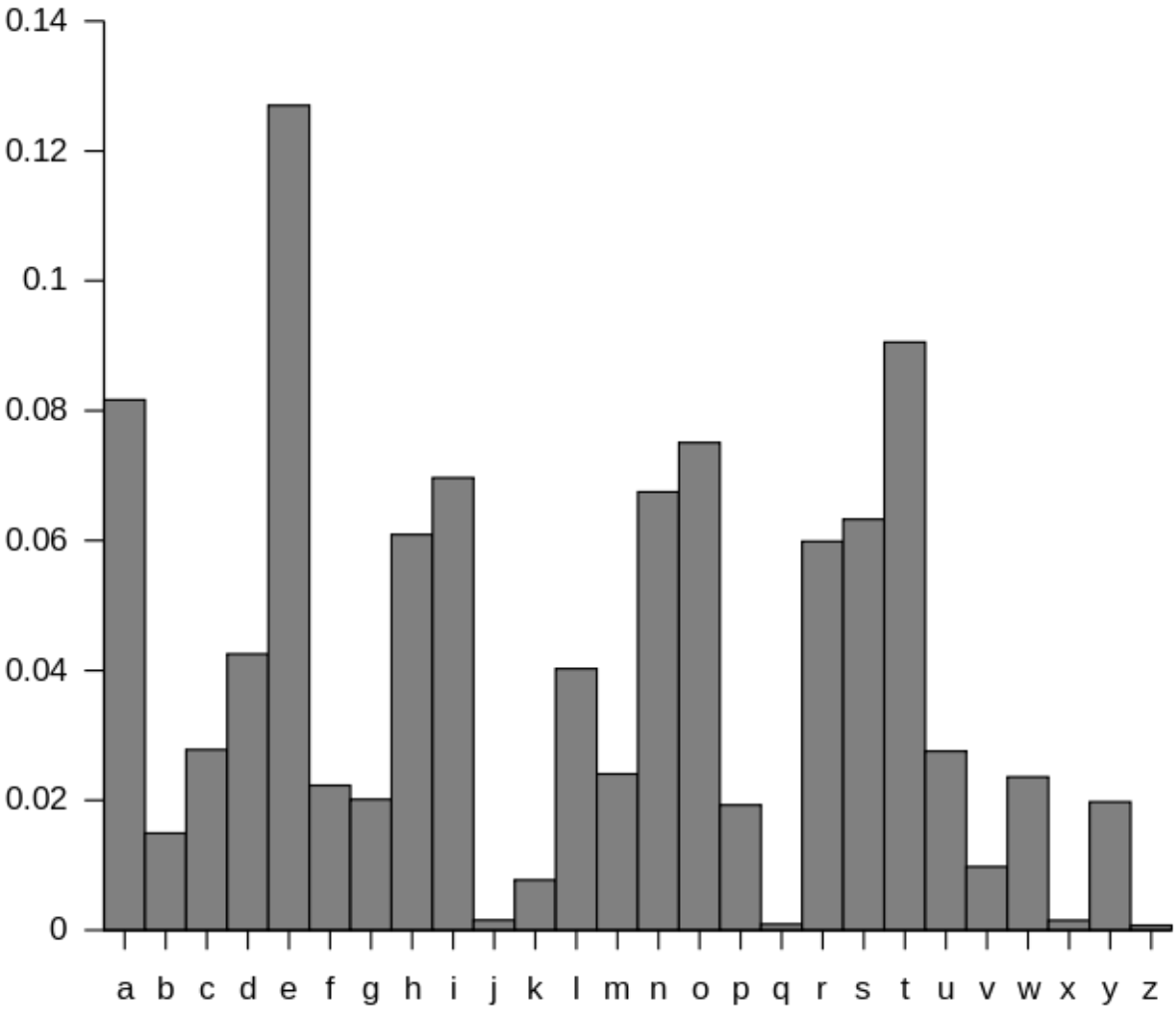
On peut appliquer la même à

```
grep -o . <<< "abcdefghijklmnopqrstuvwxyz" |
while read letter
do
  echo $letter $(grep -i $letter american-english | wc -l)
done
```

et aussi minimiser l'appel à une commande externe

```
for letter in 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
do
  echo "$letter $(grep -ic $letter american-english)"
done
```

Mais en fait, on connait déjà la fréquence des lettres discriminantes grâce à l'**analyse fréquentielle**[W] : **ETAOIN SHRDLU**[W] pour les douze premières…



Dans le cas présent, on pourrait appliquer ton algo à un dictionnaire fixe, `words5letters` obtenu par

```
grep '^.....$'  american-english
```

que j'aurais tendance à juste écrire :

```
grep -w '.....'  american-english
# ou encore
grep -Ew '.{5}'  american-english
```

Mieux, le dictionnaire fixe pourrait ne contenir que la liste mise à nue par la rétro-ingénierie de ce journal.

Dans cette liste, on pourra donc retrouver *melds* et ses anagrammes avec :

```
grep -Eiw '[sledm]{5}' words5letters
```

Et *cheap* et *peach* à partir de trois lettres seront donnés par (sauf erreur de ma part) :

```
grep -Ei '(?[chp]?){3}' words5letters
```

--

"It is seldom that liberty of any kind is lost all at once." — David Hume

Re: Autre approche : grep

Posté par **chimrod** (**site Web personnel**) le 13/01/22 à 21:34. Évalué à 4 (+2/-0).

> J'avoue ne pas avoir preté vraiment attention à la beauté des lignes de code, elles étaient davantage là pour appuyer mon propos !
>
> Je m'en sors mieux en **OCaml** (solution trouvée au quatrième coup avec le mot du jour)

Re: Autre approche : grep

Posté par **Gil Cot** le 14/01/22 à 00:42. Évalué à 4 (+2/-0).

> Pas de souci ; toute façon faire du script shell n'est pas toujours évident (raison pour laquelle dès que ça dépasse quelques lignes il vaut mieux se tourner vers un langage plus évolué que l'on maîtrise) ; et en mode exploration (usage interactif donc, comme ici) on ne se focalise pas sur la « beauté du code » (il vaut mieux d'ailleurs que ce soit les plus simple + intuitif + compréhensible…)
> J'ai voulu surtout rendre hommage à l'ami `grep` en pointant quelques raccourcis que l'on oublie souvent.
>
> Comme souvent avec OCaml, ton gist est juste beau (et on suit littéralement le raisonnement de l'arbre de décision.) Merci.
>
> --
>
> "It is seldom that liberty of any kind is lost all at once." — David Hume

Re: Autre approche : grep

Posté par **tisaac** le 16/01/22 à 17:13. Évalué à 4 (+2/-0).

> Une approche en ligne de commande mais pas vraiment optimisée est proposée **ici**.
>
> --
>
> Surtout, ne pas tout prendre au sérieux !

Re: Autre approche : grep

Posté par **Gil Cot** le 17/01/22 à 01:34. Évalué à 2 (+0/-0).

> En parcourant rapidement, l'article utilise aussi un arbre décisionnel similaire à la dichotomie s'il s'agissait de deviner un nombre. Ça part d'un dictionnaire nommé `words` et non `american-english` et crée une liste de travail appelée `myguess` (15034 entrées) et non `words5letters` comme je l'ai suggéré.

--

"It is seldom that liberty of any kind is lost all at once." — David Hume

Re: Autre approche : grep

Posté par **Gil Cot** le 20/01/22 à 21:06. Évalué à 2 (+0/-0). Dernière modification le 20/01/22 à 21:07.

Bon, votre serviteur a mis cette approche en script → **https://github.com/gilcot/wsa**
Par contre, ce n'est pas automatique comme la solution de Chimrod : le but était juste de pouvoir faire les différentes étapes de la ligne de commande avec le même script et sans trop s'y connaître en `grep`age ni créer des fichiers partout (ni faire exploser la mémoire : les listes de mots sont passées par *pipe* mais pas stockées dans une variable). Du coup, le script est *stateless* et il faut rajouter soi-même les les lettres déjà interdites aux nouvelles invalides.

D'habitude le script s'en tire en 4 coups. J'ai cependant pu obtenir juste 3 coups, une fois, dans ma dizaine d'essais. De même, pour le dernier, que je retranscrit ici, il a fallu 5 coups. Dans ces deux cas, et pour la moitié de mes tests, j'ai pris la première proposition de la liste.

```
# lancement non interactif (espace pour les arguments)
# juste histoire d'avoir une liste de selection initiale
$ ./wsa.sh -d ' ' ' ' ' '
[INFO] seeking: ..... [esiarntolcdugpmkhbyfvw] [^zxqj]
[INFO] using:   235886 /usr/share/dict/words
[INFO] scenario: nothing yet
[INFO] max results limit count: 15  Random
theow
drusy
Hokan
wield
mungy
samba
numda
kusha
pried
aweto
Vince
darky
belve
Milan
sulka
# relancer avec les indications obtenues pour restreindre
$ ./wsa.sh -d '...O.' 'T' 'HEW'
[INFO] seeking: ...o. [^t].... [t] [^hew]
[INFO] using:   235886 /usr/share/dict/words
[INFO] scenario: green yellow gray
[INFO] max results limit count: 15  Random
stoop
smoot
motor
stook
sotol
unpot
fagot
```

```
spoot
scoot
Anton
robot
gigot
idiot
untop
Ortol
# et rebelotte, et je note une correction a faire
$ ./wsa.sh -d '...O.' 'TTO..' 'SPHEW'
[INFO] seeking: ...o. [^t][^t][^o].. [tto] [^sphew]
[INFO] using:   235886 /usr/share/dict/words
[INFO] scenario: green yellow gray
[INFO] max results limit count: 15  Random
codon
adion
igloo
Conoy
Major
Modoc
manoc
cocoa
rigol
gator
ardor
ninon
cabot
baron
cabob
# rebelotte, c'est quasiment fini
$ ./wsa.sh -d '.O.O.' 'T.O..' 'CDNSPHEW'
[INFO] seeking: .o.o. [^t].[^o].. [to] [^cdnsphew]
[INFO] using:   235886 /usr/share/dict/words
[INFO] scenario: green yellow gray
[INFO] max results limit count: 15  Random
motor
fogou
Koroa
bokom
zokor
logoi
rotor
robot
# enfin, pour confirmer le mot que je pense
$ ./wsa.sh -d '.O.O.' '..T.R' 'MCDNSPHEW'
[INFO] seeking: .o.o. ..[^t].[^r] [tr] [^mcdnsphew]
[INFO] using:   235886 /usr/share/dict/words
[INFO] scenario: green yellow gray
[INFO] max results limit count: 15  Random
robot
Koroa
```

> C'est normal pour les majuscules : le dico utilisé a des noms/titres et ce test est parmi la moitié faite sans un dictionnaire épuré…
>
> --
>
> "It is seldom that liberty of any kind is lost all at once." — David Hume

Re: Autre approche : grep

Posté par **steph1978** le 21/01/22 à 17:17. Évalué à 2 (+0/-0). Dernière modification le 21/01/22 à 17:17.

> Comment prendre en compte le cas où on a deux missplaced pour la meme position ?

Re: Autre approche : grep

Posté par **Gil Cot** le 21/01/22 à 17:59. Évalué à 2 (+0/-0).

> Bien vu ! Je n'y avais pas pensé et n'avais pas eu le cas.
> Au niveau du grepage c'est assez simple. Par contre faudra probablement que je revois l'interface d'entrée.
>
> --
>
> "It is seldom that liberty of any kind is lost all at once." — David Hume

Re: Autre approche : grep

Posté par **Gil Cot** le 23/01/22 à 03:06. Évalué à 3 (+1/-0).

> Possibilité rajoutée : il faut spécifier le groupe de lettres de la position entre crochets… J'ai pu le tester sur une vraie partie en ayant eu `a..r.` puis `[ra]apr.` (respectivement pour les essais *alert* puis *rapid* ; pour trouver la solution, *spray*, au troisième coup.)
>
> --
>
> "It is seldom that liberty of any kind is lost all at once." — David Hume

lexique d'entrainement en poche

Posté par **Gil Cot** le 13/01/22 à 17:44. Évalué à 3 (+1/-0).

> *Vous proposez un mot du dictionnaire (lequel ?), le jeu vous dit quelles lettres sont dans le mot à trouver et quelles lettres sont à la bonne place, quelles lettres n'y sont pas.*
> *J'ai joué mais je ne suis pas doué. Je connais combien de mots de cinq lettres en anglais …*

Pour le vocabulaire, il faut pratiquer comme on dit.

J'étais tombé sur des jeux intéressants, de type **Boggle**, sur *smartphone droid* : **Lexic** puis **Lexica** qui prend en charge d'autres langues en plus de moderniser l'interface (par contre le code a pris un embonpoint terrible ces derniers temps…)

Un peu dans le même esprit, mais plutôt comme **Le mot le plus long**[W] sans **le célèbre plateau de Hasbro**[W], il y a **9P**

> *Par contre mon pc connaît plus de 15k mots d'anglais de cinq lettres, oui oui. Et il est capable de valider ces mots par une liste de règles (contient un "a", a un "e" à telle position, ne contient pas de "p", …). J'ai pu trouver 'favor' en quatre essais à coup de grep et grep -v. C'est donc un travail pour un ordinateur.*

Pour rester sur l'appareil de poche, en tant que cruciverbiste et un peu verbicruciste, j'étais tombé sur **Crossword** et peut-être **Regex Crosword** que je n'ai pas testé.

--

"It is seldom that liberty of any kind is lost all at once." — David Hume

suffit de demander

Posté par **Gil Cot** le 18/01/22 à 06:24. Évalué à 2 (+0/-0).

> *Exercice laissé au lecteur ou à moi même : implémenter le jeu en client-serveur afin de ne pas divulguer la liste et pourvoir tenir un leaderborad*

https://linuxfr.org/users/deuzene/liens/a-clone-of-the-popular-game-wordle-made-using-react-typescript-and-tailwind
(par contre me semble qu'il manque le *leaderboard* ou j'ai mal lu en diagonale)

--

"It is seldom that liberty of any kind is lost all at once." — David Hume