

La lecture sécurisée
31 | MAR | 2020
Dorian.H Mekni

gets()

/*

Lecture et recuperation d'une chaine de caractères

la donnée obtenue par cette action est stocké et mémorisé dans une variable



Elle n'offre pas la possibilité d'une lecture sécurisée

Redlisted

*/

atoi() | atol() | atod()

/*

Conversion sans sécurité quand on passe d'un type de donnée
à un autre

Aucun outil également de vérification



Redlisted

*/

scanf()

Elle permet l'interaction avec
l'utilisateur sur console en l'invitant à
rentrer des données

scanf() <-> fscanf()

/*

Si la donnée rentrée sur le scanf() par l'utilisateur ne correspond pas au type intégrée, cela peut causer un bug mais surtout une faille dans le programme qui peut être exploité à mauvais usage

-> Idem quant il s'agit de la lecture d'un fichier avec fscanf() <-



Utilisation

50 | 50

*/

scanf() | tips

- La fonction scanf() élimine les espaces sauf avec des types de caractères bien précis
- pour le type char que l'on précise la lettre ou bien on laisse un espace blanc, notre fonction nous retourne bien 1 même si aucun caractère n'est visible sur la console.
- pour “%c” on marquera un espace comme ceci “ %c” afin que notre fonction inclue les espaces blancs

scanf() | A

scanf() s'utilise en integrant un certain nombre de paramètres comme des conditions quant au nombre de caractères qu'une chaîne peut retourner. Un ajout de retour pour s'assurer combien d'elements sont retournés est un +

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #define NAME 6 // 6 caractère à saisir | \0 étant exclu
4
5 int main(void)
6 {
7     char name[NAME + 1]; int return1;
8     // car une chaîne de caractere possede toujours un element supplémentaire qui est le \0
9     // le \0 est la representation du caractère de fin pour une chaîne
10    printf("Word in 6 letters : ");
11    return1 = scanf("%6s", name);
12
13    printf("The chosen word is %s\n", name);
14    printf("Result = %d\n", return1);
15
16    return 0;
17 }
18
19
```



```
Word in 6 letters : dorian
The chosen word is dorian
Result = 1
Program ended with exit code: 0
```

scanf() | B

`scanf("%6s", name)` -> Ici c'est une garantie pour s'assurer de récupérer une chaîne qui ne lise uniquement 5 caractères. Dans le cas contraire, la lecture numérique ignore les autres caractères et n'en lis seulement 5.

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #define NAME 6 // 6 caractère à saisir | \0 étant exclu
4
5 int main(void)
6 {
7     char name[NAME + 1]; int return1;
8     // car une chaîne de caractère possède toujours un élément supplémentaire qui est le \0
9     // le \0 est la représentation du caractère de fin pour une chaîne
10    printf("Word in 6 letters : ");
11    return1 = scanf("%6s", name); // CONDITIONS POUR UNE LECTURE DE 6 CARACTERES
12
13    printf("The chosen word is %s\n", name);
14    printf("Result = %d\n", return1);
15
16    return 0;
17 }
18
19
```



```
Word in 6 letters : jackysing
The chosen word is jackys
Result = 1
Program ended with exit code: 0
```


scanf() | C

Voilà en mettant non seulement la limite de caractère que notre fonction prenne en compte et affiche mais aussi les lettres que nous désirons. Ici en l'occurrence le d et le r ->

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #define NAME 6 // 6 caractère à saisir | \0 étant exclu
4
5 int main(void)
6 {
7     char name[NAME + 1]; int return1;
8
9     printf("Word containing only d or r : ");
10    return1 = scanf("%6[dr]", name); // CONDITIONS POUR UNE LECTURE DE 6 CARACTERES
11
12    printf("The chosen word is %s\n", name);
13    printf("Result = %d\n", return1);
14
15    return 0;
16 }
17
18
19
```



```
Word containing only d or r : ryandfe
The chosen word is r
Result = 1
Program ended with exit code: 0
```

scanf() | D

écriture qui permet la lecture et affichage d'un nombre contenant toutes les lettres d l'alphabet

```
5 int main(void)
6 {
7     char name[NAME + 1]; int return1;
8
9     printf("Word : ");
10    return1 = scanf("%[a-z]", name);
11
12    printf("The chosen word is %s\n", name);
13    printf("Result = %d\n", return1);
14
15    return 0;
16 }
17
```



```
Word : success
The chosen word is success
Result = 1
Program ended with exit code: 0
```

scanf() | E

```
scanf("6%s", name)
```

Les doubles trémas utilisés pour éviter les débordements
tampons

fgets()

/*

fonction de lecture d'une chaîne de caractères à partir d'un
flux

Lecture sécurisée



GuestList

*/

fgets()

- Lire une chaîne de caractères à partir d'un flux
- Avec la fonction fgets(), nous pouvons sécuriser la copie des caractères de la chaîne en précisant une taille maximale à lire
- Ainsi, si la chaîne lue devient plus longue que le buffer de réception
- Il n'y aura pas de débordement en mémoire
- Le nombre de caractères sera en lecture au max de(maxLenght -1)
- Le caractère '\0' est systématiquement ajouté en fin de chaîne.

fgets()

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #define WORD_SIZE 6 // 6 caractère à saisir | \0 étant exclu
4
5 int main(void)
6 {
7     char word[WORD_SIZE + 1];
8
9     fgets(word, 7, stdin); // 7 car on inclue le caractère | 6 caractères + 1 [\0]
10    printf("%s\n", word);
11
12    return 0;
13 }
14
15
```

strtol() | A

Fonction de conversion d'un type à un autre

Fonction retournant 0 dans le cas où la conversion n'est pas possible

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #define WORD_SIZE 6 // 6 caractère à saisir | \0 étant exclu
4
5 int main(void)
6 {
7     char word[WORD_SIZE + 1];
8     long int serialNumber;
9
10    serialNumber = strtol(word, NULL, 10);
11    printf("Word -> %ld\n", serialNumber);
12    /* Dans le cas où la conversion n'est pas possible, ces valeurs vont retourner zero */
13
14    return 0;
15 }
16
17
18
```



```
Word -> 0
Program ended with exit code: 0
```

strtol() | B

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #define WORD_SIZE 6 // 6 caractère à saisir | \0 étant exclu
4
5 int main(void)
6 {
7     char word[WORD_SIZE + 1];
8     long int serialNumber;
9
10    fgets(word, 7, stdin);
11
12    serialNumber = strtol(word, NULL, 10);
13    printf("Word -> %ld\n", serialNumber);
14    /* Dans le cas où la conversion n'est pas possible, ces valeurs vont retourner zero */
15
16    return 0;
17 }
```



45678902

Word -> 456789

Program ended with exit code: 0

fflush(stdin)

/*

Fonction qui travaille sur des tampons en sortie

Fonction néanmoins indéfini donc ne peux affirmer que le
buffer d'entrée a été vidé.





Redlisted

*/

fonction manuelle -> flush

Voila une fonction montée manuellement qui permet de vider son tampon d'entrée de manière plus sécurisée que la fonction vue précédemment ->

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #define WORD_SIZE 6 // 6 caractères à saisir | \0 étant exclu
4
5
6 void flush_fonction(void) // fonction pour vider le tampon ou [buffer] de l'entrée
7 {
8     int x = 0;
9
10    while(x != '\n' && x != EOF) // EOF veut dire end of file, fin de fichier
11        x = getchar(); // getchar() est une fonction de récupération de caractère
12 }
13
14
15 int main(void)
16 {
17     char word[WORD_SIZE + 1];
18
19     printf(" Word containing 6 letters : ");
20     fgets(word, WORD_SIZE + 1, stdin);
21     printf("%s", word);
22
23     flush_fonction();
24     printf("\n");
25
26     printf(" Word containing 6 letters : ");
27     fgets(word, WORD_SIZE + 1, stdin);
28     printf("%s", word);
29
30     printf("\n");
31
32
33     return 0;
34 }
```

Word containing 6 letters : dorian
dorian
Word containing 6 letters : americ
americ
Program ended with exit code: 0

