CI 8: Chaînes de caractères

Télécom Physique Strasbourg — October 3, 2021

Traiter les exercices ci-dessous en vous inspirant des exemples du cours.

Exercice 17 *(Dialogue)*

1. Ecrire un programme qui lit le nom et prénom de l'utilisateur saisis sur une ligne avec 2 appels à `scanf()` et affiche ensuite un message de bienvenue :

```
$> loic cuvillon ↩
Bienvenue loic cuvillon dans le monde fascinant du C.
```

2. Faire de même sans utiliser `scanf()` avec:
   - un seul appel à `fgets()` qui récupére la ligne entière avec le nom et prénom;
   - un appel à `sscanf()` pour extraire le nom et prénom de la ligne, et les stocker dans 2 chaînes distinctes;
   - l'affichage du message.

Exercice 18 *(Confirmation d'un nouveau mot de passe)*

Ecrire un programme qui lit un mot de passe saisi par l'utilisateur. Puis alloue dynamiquement la mémoire nécessaire pour stocker la saisie de la confirmation du mot de passe.

Le programme indique alors si le mot de passe est confirmé ou non (`strcmp()` ou une boucle sur les caractères).

```
$> mot de passe : unchevaldanslabri ↩
$> confirmer le mot de passe: unchevaldanslelac ↩
Erreur! Mauvaise confirmation.
```

Exercice 19 *(Recherche d'un caractère dans une chaîne)*

1. Ecrire une fonction `main()` qui demande et lit successivement au clavier un caractère seul puis une phrase (`fgets()`) [a] , respectivement stockés dans un `char` et une chaîne de caractères. Afficher la chaîne et le caractère avec `printf()` pour vérifier la bonne lecture.

2. Ecrire alors une fonction qui prend en paramètre une chaîne et un caractère avant de retourner vrai(1) ou faux(0) si le caractère est présent ou non dans la chaîne.
   N'utilisez pas la fonction `strchr()` (c'est celle que l'on essaye de coder) mais une boucle de recherche.

[a] Attention aux caractères trainants dans `stdin` (la mémoire du clavier), tel le `\n`. `fgets()` s'arrête au premier rencontré et le retire de `stdin` avec les caractères qui le précèdent. Tandis que `scanf("%c",&var)` lit un seul caractère (éventuellement un `\n`) et laisse dans `stdin` le `\n` qui suit s' il existe.
Pour être sûr d'avoir `stdin` vide:
→ utiliser uniquement des `fgets()` qui lisent une ligne entière dont le `\n` final correspondant à la touche `[return]`, ou
→ vider `stdin` par une lecture en boucle jusqu'à ce qu'il soit vide.