

Programmation Concurrente

Révision langage C et make

Série 1

Exercice 1

Soit les déclarations de variables suivantes :

```
long n = 15;
```

```
long *p;
```

```
long **pp;
```

Imaginez que `p` pointe sur `n` et que `pp` pointe sur `p`.

En partant du principe que vous connaissez les adresses de `n`, `p` et `pp`, écrivez sur papier quelles seront :

- a) les valeurs de `p` et `pp` ?
- b) la valeur de `*p` ?
- c) la valeur de `*pp` ?
- d) la valeur de `**pp` ?
- e) Ecrivez un programme permettant de vérifier vos hypothèses.

Note : pour afficher un `long` utilisez le format `%ld` et pour afficher l'adresse d'une variable (i.e. pointeur), utilisez `%p`.

Exercice 2

- a) Implémentez une fonction qui remplace les espaces d'une chaîne de caractères par un caractère spécifié. Votre fonction aura comme argument une structure qui contiendra un pointeur sur le début d'une chaîne de caractères, ainsi que le caractère de remplacement des espaces. Vous devrez passer un pointeur sur la structure en argument à votre fonction. A noter que votre fonction ne retournera rien. **Attention** : assurez-vous de passer un pointeur sur un tableau de caractères non constant à votre fonction, car si vous essayez d'écrire dans une chaîne de caractères littérale, votre programme risque de crasher ! Par exemple écrire `f("ma chaine")` passe une chaîne de caractères qui est constante à la fonction `f()`.
- b) Ecrivez ensuite un programme de test permettant de vérifier le bon fonctionnement de votre implémentation.
- c) Enfin, écrivez un makefile permettant de compiler votre code, en incluant une cible `clean` permettant de faire le ménage.

Exercice 3

- Implémentez une fonction similaire à celle de l'exercice précédent, mais qui va en plus inverser la chaîne de caractères. Cette fois-ci, on ne désire pas modifier la chaîne de caractères passée en argument, mais **renvoyer** une nouvelle chaîne, inversée avec les espaces remplacés par le caractère spécifié. La nouvelle chaîne renvoyée sera allouée avec la fonction `malloc`.
- Ecrivez ensuite un programme de test afin de vérifier que votre code fonctionne correctement.
- Enfin, écrivez un makefile permettant de compiler votre code, en incluant une cible `clean` permettant de faire le ménage.

Exercice 4

Implémentez un programme qui prend les arguments suivants en ligne de commande :

"label" <n (nombre itérations)> <x (nombre décimal)> <y (nombre entier) >

Le programme affichera le label, puis la n fois $x^{(y+m)}$ où m appartient à $\{0, 1, 2, \dots, n-1\}$

Le nombre d'arguments doit être exactement de 4, sinon le programme doit afficher une erreur (qui peut être son mode d'emploi par exemple).

Note : ajoutez l'option `-lm` à la ligne de compilation pour indiquer au compilateur que la librairie mathématique est utilisée

Exemple :

```
./ex4 calcul: 5 3.4 2
```

calcul:

$3.400000^2 = 11.560001$

$3.400000^3 = 39.304003$

$3.400000^4 = 133.633615$

$3.400000^5 = 454.354304$

$3.400000^6 = 1544.804676$