POLYTECH LILLE

Université Lille1 Sciences et Technologies

Département IMA / 3A (S5) **Programmation Structurée 2010/2011**

TP Chaînes de caractères et pointeurs

Objectifs

Manipuler les entrées/sorties C (via les redirections), implémenter en C quelques algorithmes de chaînes de caractères, utiliser les fonctions de `string.h`.

1 Redirections et Entrées/Sorties

On fournit dans le fichier `tpchaines.c` une procédure et un programme principal :

```
#include <stdio.h>
#include <stdbool.h>

void mange_et_reecrit(void)
{
  char c;
  c = getchar();
  while ( c != EOF )
    {
      putchar(c);
      c = getchar();
    }
}

int main(void)
{
  mange_et_reecrit();
  return 0;
}
```

EXERCICE 1 *On demande de compiler le programme et de l'exécuter :*

- *Que fait ce programme ?*
- *Écrire un fichier de nom* `toto` *(sans extension, ce n'est pas un programme), qui contient :*
  `truc`
  `bidule`
  `plouf`
  (pas de saut de ligne après `plouf`*).*
- *Dans le terminal, faire* `copy <toto`*. Que ce passe-t-il ?*
- *Que va-t-il se passer si on fait* `copy <toto >titi` *? Vérifier. Comparer avec la commande* `cp` *d'Unix.*

Remarque : on pourrait utiliser scanf au lieu de getchar.

2 Algorithmes de chaînes

Pour les deux exercices suivants, on écrira les programmes sur le modèle précédent (**un .c par exercice**).

Exercice 2 *Écrire un programme qui compte le nombre de caractères (y compris les espaces, les sauts de lignes, les tabulations) de l'entrée standard. Utiliser les redirections pour compter le nombre de charactères du fichier* `toto`.

Exercice 3 *Écrire un programme qui imprime les lettres qui ne sont pas apparues dans le fichier (on suppose que les lettres sont uniquement des minuscules).*
Indication : On pourra utiliser un tableau de 26 *cases que l'on mettra à jour à chaque fois que l'on rencontrera une lettre. Comment trouver l'indice de la case à mettre à jour ?*

3 Fonctions de la lib standard pour les chaînes

Les exercices de cette section seront réalisés avec l'aide des fonctions de la librairie `string.h`. Les chaînes de caractères auront une taille statique déclarée à l'avance : `char ch1[N],ch2[N]` lorsque c'est possible. **Créez un nouveau .c pour cette section**

Exercice 4 *À l'aide de* `fgets` *(*`man 3 fgets`*), demander une chaîne de caractères à l'utilisateur, et imprimer la taille de celle-ci (*`strlen`*). Remarquer que* `fgets` *récupère aussi le saut de ligne, et écrire une fonction* `char* monfgets()` *qui retourne la chaîne donnée par l'utilisateur, sans le saut de ligne.*

Exercice 5 *À l'aide de la fonction précédente, demander deux chaînes de charactères à l'utilisateur (de taille max* $NMAX$ *constante). Utiliser* `strcat` *pour concaténer ces deux chaînes et afficher le résultat. La chaîne résultat est forcément un pointeur.*

Exercice 6 *Écrire une fonction qui détermine si deux chaînes de caractères sont préfixes l'une de l'autre, en utilisant* `strcmp`. *Tester.*

Exercice 7 *Écrire une fonction qui détermine si une chaîne de caractères est une sous-chaîne d'une deuxième. On utilisera* `index` *et un pointeur pour parcourir la deuxième chaîne.*

4 Si vous avez encore du temps

Exercice 8 *Écrire une fonction qui prend en argument deux chaînes de caractères (sous forme de tableaux de caractères de taille* `MAX` *dans lesquels les fins de chaînes sont repérées par* `'\0'`*) et qui dit si ces deux chaînes sont anagrammes l'unes de l'autre. Par exemple,* `marine` *et* `irmane` *sont anagrammes.*

Exercice 9 *Écrire un programme qui écrit le fichier à l'envers. Sur le fichier* `toto`*, le programme donnera :*

```
fuolp
eludib
curt
```

Exercice 10 *Écrire un programme qui dit si les parenthèses d'un fichier sont bien imbriquées. On ignore les autres caractères. Dans* `(())()` *les parenthèses sont bien imbriquées, mais pas dans* `((((()(`.