

## Chap. 5: Les chaînes de caractères

# Le type chaîne

- ▶ Pas de type *string* en C !
- ▶ Mais des **tableaux de caractères**
- ▶ Spécificité:  
terminés par un caractère spécial: **'\0'** (0X00)
- ▶ Littéraux: caractères encadrés du caractère **"**

```
char *str = "Hello world\n"
```



# Déclaration de chaînes

3

1) chaîne en tant que **tableau** de caractères:

```
char s1[25] = "Hello!"; // taille = 25  
char s2[] = "Hello!";   // taille = 7
```

⇒ espace mémoire réservé à la compilation

2) chaîne en tant que **pointeur** sur un caractère:

```
char *s2;
```

⇒ aucune mémoire n'est allouée  
pour stocker la chaîne!

# Affectation de chaînes

4

- 1) chaîne en tant que **tableau** de caractères:

```
char s1[25] = "Hello!";  
s1[0] = 'A'; s1[1] = '\\0';
```

- 2) chaîne en tant que **pointeur** sur un caractère:

```
char *s2;  
s2 = "Hello!";    // littéral  
s2 = (char*) malloc(25*sizeof(char));  
s2 = s1;          // où s1 est un tableau de char  
s2 = s3;          // où s3 est un pointeur  
                  // vers un char
```

## ► Lecture de chaîne sur *stdin*:

```
char* fgets (char *s, int size, FILE *stream)
```

`fgets()` reads in at most one less than *size* characters from *stream* and stores them into the buffer pointed to by *s*.

Reading stops after an EOF or a newline. If a newline is read, it is stored into the buffer. A terminating null byte ('\0') is stored after the last character in the buffer.

`fgets()` returns *s* on success, and NULL on error or when end of file occurs while no characters have been read.

# Ecriture de chaîne

- Ecriture de chaînes sur *stdout*:

```
int printf (const char* format, ...)
```

`printf()` writes the output under the control of a *format* string that specifies how subsequent arguments are converted for output.

# Lecture/Ecriture de chaînes

7

Exemple:

```
char ligne[256];  
while (fgets(ligne, 256, stdin) != NULL) {  
    printf("La chaîne lue est '%s'", ligne);  
}
```

- ⇒ lecture sur *stdin* de lignes de maximum 255 caractères (dont – éventuellement – le passage à la ligne '\n') et ajout de '\0'
- ⇒ écriture sur *stdout* de la chaîne lue

# Traitement des chaînes

- ▶ Propriétés des tableaux
- ▶ Espace mémoire réservé:
  - à la compilation (tableaux de caractères)
  - par allocation dynamique (pointeur sur un caractère)
- ▶ Pas de stockage de la taille de la chaîne  
→ information gérée par l'application
- ▶ S'assurer que la taille est suffisante pour stocker tous les caractères + le caractère '`\0`'



# Fonctions standard

- ▶ Pas de types string  $\Rightarrow$  pas d'opérateurs!
- ▶ Fonctions standard définies dans **string.h**

- ❑ `size_t strlen (const char* s);`
- ❑ `char* strcpy (char* dest, const char* src);`
- ❑ `char* strcat (char* dest, const char* src);`
- ❑ `int strcmp (const char* dest, const char* src);`

- + fonctions de base avec taille maximale
- + fonctions de recherche
- + fonctions de conversions numériques
- + fonctions de traitement sur la mémoire

# Copie de chaîne

10

- Copie d'une chaîne dans un buffer :

```
char* strcpy (char *dest, char *src)
```

The `strcpy()` function copies the string pointed to by *src*, including the terminating null byte (`'\0'`), to the buffer pointed to by *dest*. The strings may not overlap, and the destination string *dest* must be large enough to receive the copy.

## ► Macros définies dans le fichier d'entête **cctype.h**

❑ `int isalpha(int c)`

❑ `int islower(int c)`

❑ `int isupper(int c)`

❑ `int isdigit(int c)`

❑ `int isalnum(int c)`

❑ `int isxdigit(int c)`

❑ `int ispunct(int c)`

❑ `int isprint(int c)`

❑ `int isgraph(int c)`

❑ `int iscntrl(int c)`

❑ `int isspace(int c)`

❑ `int isascii(int c)`

❑ `int tolower(int c)`

❑ `int toupper(int c)`

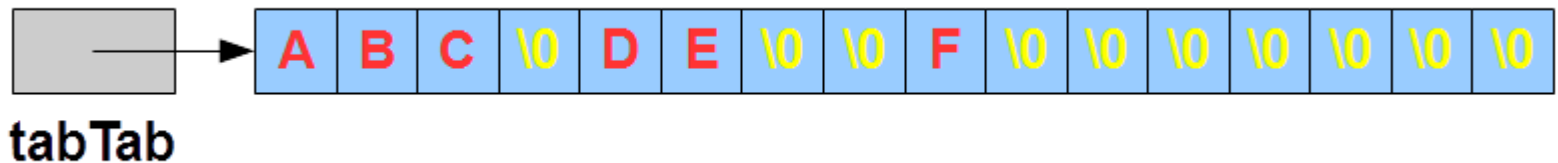
# Tableaux de chaînes

12

1) Table définie à la compilation:

- Table de tables de caractères

```
char tabTab[4][4] = {"ABC", "DE", "F"};
```

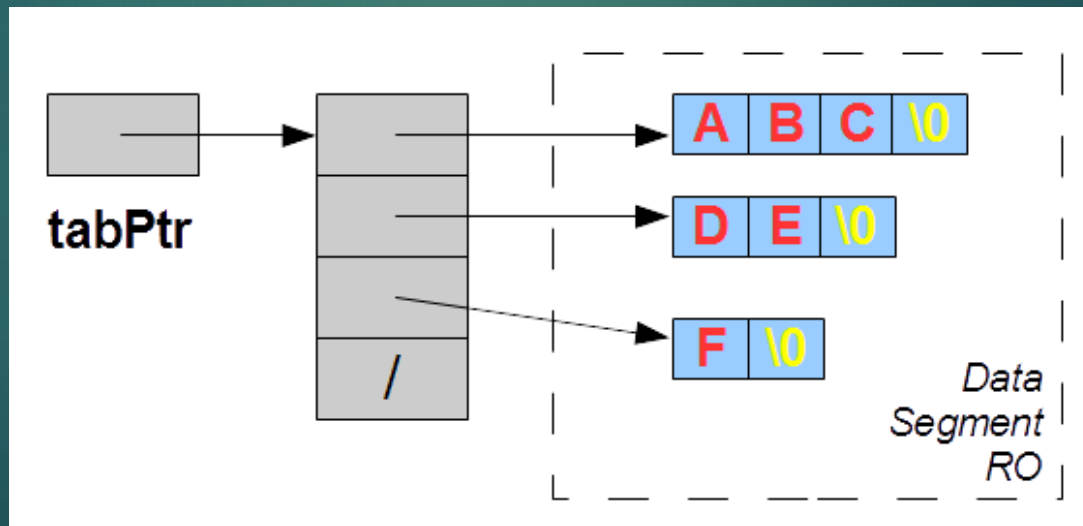


# Tableaux de chaînes

## 1) Table définie à la compilation:

- Table de pointeurs sur un caractère

```
char* tabPtr[4] = {"ABC", "DE", "F"};
```

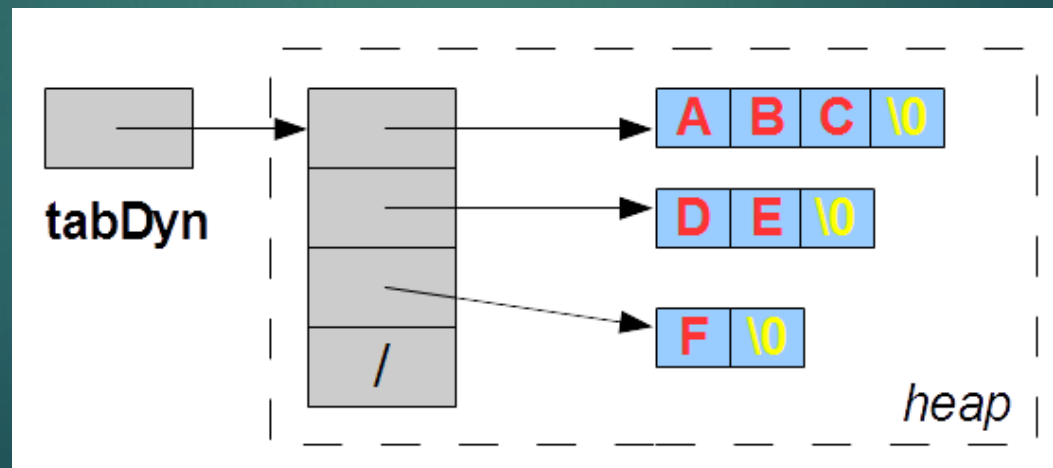


# Tableaux de chaînes

14

## 2) Table dynamique:

```
char **tabDyn = (char**)malloc(4*sizeof(char*));  
for (int i=0; i<4; i++) {  
    ...  
    tabDyn[i] = (char*)malloc((n+1)*sizeof(char));  
    ...  
}
```



# Cas particulier

15

## ► Tableau des arguments du programme

```
int main (int argc, char *argv[]) {  
    ...  
}
```

