



SL124_LOGA Programmation en C

Théorie

Chapitre 9

Les chaînes de caractères en C

Christian HUBER (CHR)

Version 1.5 Mai 2012

Version 1.6 Avril 2014

CONTENU DU CHAPITRE 9

9.	<i>Les chaînes de caractères en C</i>	9-1
9.1.	Objectifs	9-1
9.2.	La nature d'une chaîne de caractère	9-1
9.2.1.	Chaînes non modifiables	9-1
9.2.2.	Chaînes modifiables	9-2
9.2.3.	Illustration de la nature des chaînes	9-2
9.3.	Déclaration et initialisation d'une chaîne	9-3
9.3.1.	Cas des chaînes non modifiables	9-3
9.3.2.	Cas des chaînes modifiables	9-3
9.4.	Copies de chaînes	9-4
9.4.1.	strcpy, exemple	9-4
9.4.2.	strcpy, exemple, résultat	9-5
9.5.	Concaténation de chaînes	9-6
9.5.1.	strcat, exemple	9-6
9.5.2.	strcat, exemple, résultat	9-7
9.6.	Comparaison de chaînes	9-8
9.6.1.	strcmp, exemple	9-8
9.7.	Recherche dans une chaîne	9-9
9.7.1.	strstr, exemple	9-9
9.8.	Conclusion	9-10
9.9.	Historique des versions	9-10
9.9.1.	Version 1.2 Juin 2008	9-10
9.9.2.	Version 1.3 Mai 2010	9-10
9.9.3.	Version 1.4 Mai 2011	9-10
9.9.4.	Version 1.5 Mai 2012	9-10
9.9.5.	Version 1.6 Avril 2014	9-10

9. LES CHAINES DE CARACTERES EN C

Ce chapitre a pour but de regrouper tout ce qui a déjà été présenté au niveau des chaînes de caractères et de leur utilisation. Ceci afin d'avoir une vision globale de la manipulation des chaînes de caractères appelées *string* en anglais.

9.1. OBJECTIFS

A la fin de ce chapitre, l'étudiant sera capable de :

- Déclarer et initialiser une chaîne de caractères
- De copier des chaînes de caractères
- De passer des chaînes en paramètres à des fonctions
- D'utiliser quelques fonctions de manipulation

9.2. LA NATURE D'UNE CHAINE DE CARACTERE

Vu sous l'aspect du contenu mémoire une chaîne de caractère est un tableau d'éléments de type char, suivit d'un code de fin représenté par la valeur numérique 0 (zéro).

Les chaînes de caractères peuvent être modifiables ou non modifiable. Les chaînes modifiables sont stockées dans la mémoire donnée, les non modifiable dans la mémoire du programme.

9.2.1. CHAINES NON MODIFIABLES

Une chaîne de caractères non modifiables est indiquée comme **const char *** dans le prototype d'une fonction.

Au niveau de l'utilisation il est possible d'utiliser directement une chaîne entre guillemet, comme dans l'exemple suivant :

```
printf ("Hello");
```

Si on veut une constante nommée on utilisera :

```
char *s1 = "Hello";
```

ou mieux :

```
const char *s1 = "Hello";
```

La chaîne est utilisée de la manière suivante :

```
printf ("%s", s1);
```

On a donc utilisé un pointeur sur un char, que l'on initialise, et cela est considéré comme une chaîne de caractère non modifiable.

9.2.2. CHAINES MODIFIABLES

Une chaîne modifiable est déclarée comme étant un tableau de caractères. Cependant au niveau du type utilisé dans une fonction on utilise **char ***

Par exemple

```
char Chaine2[20];
```

Il est aussi possible de la déclarer et de l'initialiser.

```
char Chaine2[20] = "Bonjours";
```

Si on initialise directement, il n'y a pas besoin de dimension.

```
char Chaine2[] = "Bonjours";
```

9.2.3. ILLUSTRATION DE LA NATURES DES CHAINES

Prenons un exemple tout d'abord, avec deux façons différentes de déclarer et d'initialiser la chaîne.

```
// Déclaration et initialisation de chaînes
const char *Chaine1 = "Hello";          // non modifiable
char Chaine2[20] = "Bonjours";          // modifiable
```

Dans les deux cas, avec `chaine1` et `chaine2` nous donnerons l'adresse du tableau en mémoire. En utilisant comme ci dessous une boucle d'affichage et en utilisant le pointeur `pTmp`.

```
// Directives de compilation
#include <stdio.h>                      // pour printf

// Programme principal
int main(void)
{
    char *Chaine1 = "Hello";
    char Chaine2[20] = "Bonjours";
    char *pTmp;

    // Affichage du contenu de la chaîne
    printf ("Adresse Chaine 1 %08X \n", Chaine1);
    pTmp = Chaine1;
    while ( *pTmp != 0 ) {
        printf ("Val hexa du char %02X  val ASCII %C \n",
                *pTmp, *pTmp);
        pTmp++;
    }
    printf ("Adresse Chaine 2 %08X \n", &Chaine2[0]);
    pTmp = Chaine2;
    while ( *pTmp != 0 ) {
        printf ("Val hexa du char %02X  val ASCII %C \n",
                *pTmp, *pTmp);
        pTmp++;
    }
    return (0);
}
```

Nous pouvons obtenir l'adresse et le contenu de la mémoire. Le caractère servant de critère de fin de boucle n'est pas affiché, mais il est bien présent à la suite.

```

C:\ "d:\h\etcourssw\loga_c\théorie_1\projcours\chap...
Adresse Chaîne 1 00424058
Val hexa du char 48 val ASCII H
Val hexa du char 65 val ASCII e
Val hexa du char 6C val ASCII l
Val hexa du char 6C val ASCII l
Val hexa du char 6F val ASCII o
Adresse Chaîne 2 0012FEC0
Val hexa du char 42 val ASCII B
Val hexa du char 6F val ASCII o
Val hexa du char 6E val ASCII n
Val hexa du char 6A val ASCII j
Val hexa du char 6F val ASCII o
Val hexa du char 75 val ASCII u
Val hexa du char 72 val ASCII r
Val hexa du char 73 val ASCII s
Press any key to continue
  
```

On remarque que l'adresse des deux chaînes est très différente, Chaîne1 se trouve dans une zone de donnée non modifiable, chaîne2 elle se situe dans une zone modifiable.

9.3. DECLARATION ET INITIALISATION D'UNE CHAÎNE

Le problème avec les chaînes de caractères c'est la nécessité de les initialiser en même temps qu'on les déclare, surtout avec la forme tableau.

9.3.1. CAS DES CHAINES NON MODIFIABLES

Dans le cas des chaînes non modifiables, il est possible de déclarer et d'initialiser séparément. Par exemple :

```

const char *Chaîne1;           // déclaration

Chaîne1 = "Hello";             // initialisation
  
```

9.3.2. CAS DES CHAINES MODIFIABLES

Dans le cas d'une chaîne modifiable, il n'est pas possible d'initialiser séparément après la déclaration.

```

char Chaîne2[20];
Chaîne2 = "Bonjours";
error C2106: '=' : l'opérande gauche doit être une l-value
  
```

Si on souhaite utiliser la déclaration sous forme de tableau, donc modifiable, il est nécessaire d'utiliser une fonction de copie:

```

#include <string.h>           // pour strcpy

strcpy (Chaîne2, "Bonjours");
  
```

9.4. COPIES DE CHAINES

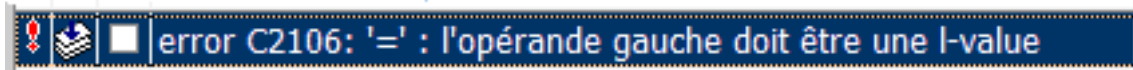
Il est nécessaire aussi d'utiliser la fonction **strcpy** pour copier une **chaîne** dans une autre. Car avec la situation suivante :

```
char Chaine3[20] = "Bonsoir";
char Chaine4[20];
```

Si on écrit :

```
Chaine4 = Chaine3;
```

Le compilateur nous indique :



Le prototype de la fonction **strcpy** est :

```
char *strcpy (char *s1, const char *s2);
```

La chaîne s1 doit être une chaîne modifiable (déclaration sous forme de tableau de caractères). La chaîne s2 est indiquée comme non modifiable, peut être fournie sous la forme d'une chaîne entre guillemet ou du nom du pointeur.

La valeur retournée correspond à s1 après la copie.

La copie est réalisée caractère par caractères, le caractère de fin de chaînes est aussi copié.

9.4.1. STRCPY, EXEMPLE

```
#include <stdio.h> // pour printf
#include <string.h> // pour strcpy

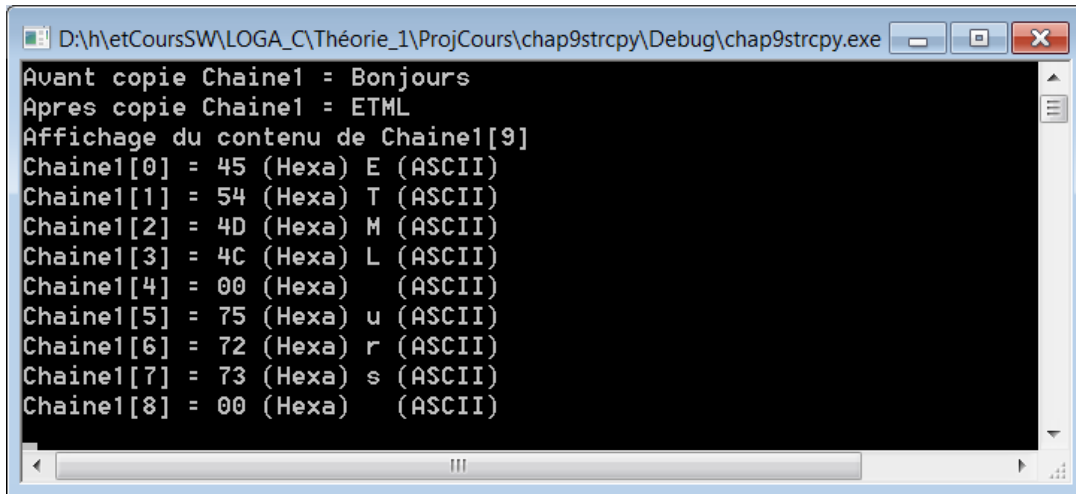
// Programme principal
int main(void)
{
    char val, i;
    char *Chaine2 = "ETML";
    char Chainel[9] = "Bonjours";
    // Affichage Chainel avant copie
    printf ("Avant copie Chainel = %s \n", Chainel);

    strcpy(Chainel, Chaine2);          // Copie

    // Affichage Chainel après copie
    printf ("Après copie Chainel = %s \n" , Chainel);

    // Affichage du détail du contenu du tableau
    printf ("Affichage du contenu de Chainel[9] \n");
    for (i=0; i < 9; i++) {
        val = Chainel[i];
        printf ("Chainel[%d] = %02X (Hexa) %C (ASCII) \n" ,
                i, val, val);
    }
    return (0);
}
```


9.4.2. STRCPY, EXEMPLE, RESULTAT



```
D:\h\etCoursSW\LOGA_C\Théorie_1\ProjCours\chap9strcpy\Debug\chap9strcpy.exe
Avant copie Chaine1 = Bonjours
Apres copie Chaine1 = ETML
Affichage du contenu de Chaine1[9]
Chaine1[0] = 45 (Hexa) E (ASCII)
Chaine1[1] = 54 (Hexa) T (ASCII)
Chaine1[2] = 4D (Hexa) M (ASCII)
Chaine1[3] = 4C (Hexa) L (ASCII)
Chaine1[4] = 00 (Hexa) (ASCII)
Chaine1[5] = 75 (Hexa) u (ASCII)
Chaine1[6] = 72 (Hexa) r (ASCII)
Chaine1[7] = 73 (Hexa) s (ASCII)
Chaine1[8] = 00 (Hexa) (ASCII)
```

Au niveau affichage avec le `printf ("Avant copie Chaine1 = %s \n", Chaine1);`
On obtient "Bonjours"

Au niveau affichage avec le `printf ("Apres copie Chaine1 = %s \n", Chaine1);`
On obtient "ETML"

Au niveau du détail du contenu de Chaine1[9], on constate que la copie a remplacé l'ancienne valeur de la chaîne (Bonjours) par la nouvelle (ETML). Le caractère de fin de chaîne est maintenant en [4]. La fin de l'ancienne chaîne n'est pas modifiée, mais elle n'est plus exploitable.

9.5. CONCATENATION DE CHAINES

La fonction **strcat** ajoute une copie de la chaîne s2 à la chaîne s1.

```
char *strcat (char *s1, const char *s2);
```

La chaîne s1 doit être une chaîne modifiable (déclaration sous forme de tableau de caractères). La chaîne s2 est indiquée comme non modifiable, peut être fournie sous la forme d'une chaîne entre guillemet ou du nom du pointeur.

La valeur retournée correspond à s1 après la concatenation.

9.5.1. STRCAT, EXEMPLE

```
#include <stdio.h> // pour printf
#include <string.h> // pour strcat

// Programme principal
int main(void)
{
    char val, i;
    char *Chaine2 = "SIDA";
    char Chaine1[9] = "STOP";
    char Chaine1[9] = "STOP";
    char *Chaine3;      // pour test valeur retour de strcat

    printf("Chaine1 = %s \n", Chaine1);
    printf("Chaine2 = %s \n", Chaine2);

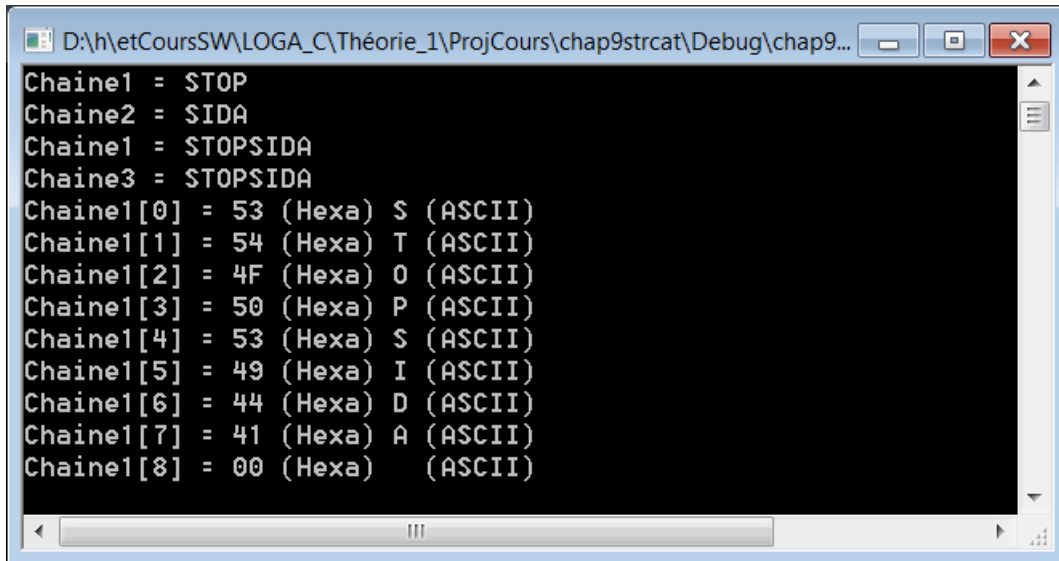
    // Concatenation
    Chaine3 = strcat(Chaine1, Chaine2);

    printf("Chaine1 = %s \n", Chaine1);
    printf("Chaine3 = %s \n", Chaine3);

    // Affichage du contenu de la chaîne
    // après concaténation
    for (i=0; i < 9; i++) {
        val = Chaine1[i];
        printf ("Chaine1[%d] = %02X (Hexa) %C (ASCII) \n",
                i, val, val);
    }
    return (0);
}
```

Remarque : il est possible de ne pas utiliser la valeur de retour de **strcat**.

```
strcat(Chaine1, Chaine2);
```

9.5.2. STRCAT, EXEMPLE, RESULTAT

```
Chaine1 = STOP
Chaine2 = SIDA
Chaine1 = STOPSIDA
Chaine3 = STOPSIDA
Chaine1[0] = 53 (Hexa) S (ASCII)
Chaine1[1] = 54 (Hexa) T (ASCII)
Chaine1[2] = 4F (Hexa) O (ASCII)
Chaine1[3] = 50 (Hexa) P (ASCII)
Chaine1[4] = 53 (Hexa) S (ASCII)
Chaine1[5] = 49 (Hexa) I (ASCII)
Chaine1[6] = 44 (Hexa) D (ASCII)
Chaine1[7] = 41 (Hexa) A (ASCII)
Chaine1[8] = 00 (Hexa) (ASCII)
```

On constate l'ajout de la chaîne "SIDA" à la suite du "STOP".

Chaine1 est modifiée. On obtient le même résultat dans chaine3.

Au niveau de l'observation détaillée, on observe la création d'une chaîne unique avec le Null déplacé à la fin de la nouvelle chaîne.

9.6. COMPARAISON DE CHAINES

La fonction **strcmp** compare la chaîne s1 avec la chaîne de référence s2.

```
int strcmp (char *s1, const char *s2);
```

La chaîne s1 est en principe une chaîne modifiable.

La chaîne s2 est indiquée comme non modifiable, peut être fournie sous la forme d'une chaîne entre guillemet ou du nom du pointeur.

La valeur retournée est:

- 0 si s1 est identique à s2;
- >0 si la chaîne s1 est plus longue que s2;
- <0 si la chaîne s1 est plus courte que s2.

9.6.1. STRCMP, EXEMPLE

```
#include <stdio.h> // pour printf
#include <string.h> // pour strcmp

// Programme principal
int main(void)
{
    int cmpRes;
    char Chainel[7] = "LOGA_C";

    // Comparaison
    cmpRes = strcmp(Chainel, "LOGA_C");
    if (cmpRes == 0 ) {
        printf( "Chainel = LOGA_C \n");
    }
    cmpRes = strcmp(Chainel, "LOGA_VB");
    if (cmpRes != 0 ) {
        printf("Chainel <> LOGA_C, cmpRes = %d \n",
               cmpRes);
    }

    return (0);
}
```



9.7. RECHERCHE DANS UNE CHAÎNE

La fonction **strstr** recherche la 1^{ère} occurrence de la chaîne s2 dans la chaîne s1.

```
char *strstr (char *s1, const char *s2);
```

La chaîne s1 est en principe une chaîne modifiable.

La chaîne s2 est indiquée comme non modifiable, peut être fournie sous la forme d'une chaîne entre guillemet ou du nom du pointeur.

La valeur retournée est un pointeur sur la chaîne localisée dans s1. Si la chaîne n'est pas trouvée la valeur retournée est NULL.

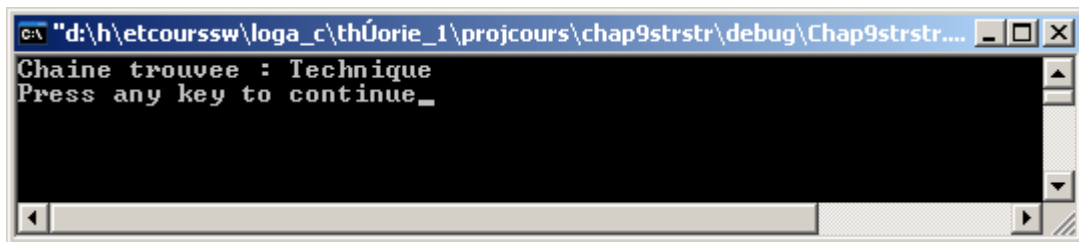
9.7.1. STRSTR, EXEMPLE

```
#include <stdio.h> // pour printf
#include <string.h> // pour strstr

// Programme principal
int main(void)
{
    char Chainel[20] = "Ecole Technique";
    char *strRes;

    // Recherche
    strRes = strstr(Chainel, "Tech");
    if (strRes != NULL) {
        printf( "Chaine trouvee : %s \n", strRes);
    }

    return (0);
}
```



9.8. CONCLUSION

Ce chapitre a présenté les principes de gestion des chaînes de caractères, ainsi que certaines des fonctions de manipulation de chaîne de caractères que l'on trouve dans **string.h**.

9.9. HISTORIQUE DES VERSION

9.9.1. VERSION 1.2 JUIN 2008

Version adaptée au Visual studio 2005

9.9.2. VERSION 1.3 MAI 2010

Introduction historique et adaptation à office 2007

9.9.3. VERSION 1.4 MAI 2011

Changement de Logo.

9.9.4. VERSION 1.5 MAI 2012

Complément des exemples strcpy et strcat.

9.9.5. VERSION 1.6 AVRIL 2014

Ajout numéro du module.