

```

// cette fonction est une implémentation de la fonction strcmp en C
// ASTUCE : man strcmp

// la fonction strcmp compare les 2 chaines de caractères s1 et s2
// cette comparaison est effectuée en utilisant des "unsigned characters"
// (en comparant les caractères de la chaine après les avoir convertis en unsigned char)
// strcmp doit retourner un entier indiquant le résultat de la comparaison :
//      0 si s1 et s2 sont identiques
//      une valeur négative si s1 est inférieure à s2
//      une valeur positive si s1 est supérieure à s2

// la valeur de retour est égale au résultat arithmétique de la
// différence entre les 2 premiers octets (caractères)
// (chacun interprétés comme de type unsigned char)
// qui diffèrent dans les chaines de caractères comparées

// autrement dit, il faut comparer les caractères de s1 et de s2
// deux à deux (le premier caractère de s1 est comparé au 1er caractère de s2,
// puis le deuxième de s1 est comparé au deuxième caractère de s2, etc)
// et, dès que le même caractère de s1 diffère du même caractère de s2,
// il faut effectuer la soustraction de ce caractère de s1 par ce caractère de s2
// (en fait, la soustraction de leur code décimal sera effectuée)

// ATTENTION : il faut convertir les paires de caractères (de s1 et de s2)
// en unsigned char avant leur comparaison et leur soustraction

// ft_strcmp retourne un int (le résultat de la soustraction
// de la première paire de caractères différente entre s1 et s2)
// elle prend en paramètres les 2 chaines de caractères s1 et s2
// (sous la forme d'un pointeur vers des char)

int    ft_strcmp(char *s1, char *s2)
{
    // - tant que la chaine de caractère s1 n'a pas atteint sa fin :
    // tant que le caractère actuel de s1 (la valeur à l'adresse s1)
    // n'est pas le caractère nul (n'est pas nulle))
    // la condition (*s1) signifie : tant que *s1 n'est pas nul
    // (la valeur en décimal du caractère nul '\0' est 0 !)

```

```

// ET (&&) - tant que la chaine de caractère s2 n'a pas atteint sa fin
// REMARQUE : avec cette condition, on sortira de la boucle
// dès que l'une des chaines de caractères aura atteint sa fin
// cela n'altère pas le fonctionnement attendu de la fonction (voir *)

// ET (&&) - tant que les caractères actuels de s1 et de s2
// (convertis en unsigned char) sont égaux

// on sortira donc de la boucle lorsqu'au moins une des chaines se sera terminée
// (s1, ou s2, ou les deux en même temps)
// ou que des caractères différents seront trouvés au même niveau
while ((*s1 && *s2) && ((unsigned char)*s1 == (unsigned char)*s2))
{
    // on passe au
    // caractère suivant de chaque chaine (pour comparer les caractères
    // de s1 et de s2 deux à deux, par paire)
    s1++;
    s2++;
}

// on retourne la différence entre les caractères actuels de s1 et s2
// (après leur conversion en unsigned char)
return ((unsigned char)*s1 == (unsigned char)*s2);
}

// RESULTAT :
// 3 cas :
// 2 caractères différents ont été trouvés au même niveau :
// *s1   -   *s2
// b     -   a
// 98    -   97
// => 1
// s1 > s2

// *s1   -   *s2
// a     -   b
// 97    -   98

```

```

// => -1
// s1 < s2

// tous les caractères sont identiques
// (on a atteint la fin des 2 chaines en même temps) :
// *s1 - *s2
// 0 - 0
// => 0
// s1 == s2

// une chaine est plus courte que l'autre
// (on a atteint la fin d'une chaine avant l'autre) :
// s1 plus courte que s2 :
// *s1 - *s2
// 0 - a
// 0 - 97
// => -97
// s1 < s2

// s1 plus longue que s2 :
// *s1 - *s2
// a - 0
// 97 - 0
// => 97
// s1 > s2

```

```

#include "ft_strcpy.h"
#include "ft_strcmp.h"
#include "ft_putnbr.h"

```

```

int    main(void)
{
    char    s1[4];
    char    s2[4];
    int      cmp;

    ft_strcpy(s1, "XYZ");
    ft_strcpy(s2, "ABC");

```

```
    cmp = ft_strcmp(s1, s2);  
    ft_putnbr(cmp);  
    return (0);  
}
```