```
// cette fonction est une implémentation de la fonction strstr en C
// ASTUCE : man strstr
// la fonction strstr localise une sous-chaine dans une chaine de caractères

// cette fonction trouve la première occurrence de la sous-chaine needle
// (to_find dans notre fonction ft_strstr) dans la chaine de caractères haystack
// (str dans notre fonction ft_strstr)
// les caractères de fin de chaine '\0' ne sont pas comparés

// la valeur de retour est un pointeur vers le début de la sous-chaine trouvée dans str
// (si tel est le cas), ou NULL si la sous-chaine n'a pas été trouvée

// si needle (to_find) est une chaine vide, la valeur de retour sera haystack (str)

// pour cela :
// - on vérifie tout d'abord si to_find est une chaine de vide : si c'est le cas,
// on retourne immédiatement str

// - on stocke l'adresse du début de to_find dans to_find_start
// (l'adresse vers le premier caractère à chercher dans str, "pointeur de recherche")
// (pour revenir au premier caractère de to_find, dans le cas où to_find n'a été trouvé
// que partiellement dans str (pour pouvoir relancer la recherche depuis le début
// de la sous-chaine à chercher))
// Exemple : to_find_start = adresse de n (dans to_find)

// - tant que la fin de la chaine str n'est pas atteinte
// (tant que toute la chaine str n'a pas été parcourue)
// (pour parcourir la chaine jusqu'à la fin, sans comparer les caractères de fin de chaine,
// et pour pouvoir gérer le cas où la sous-chaine n'a pas été trouvée, pour retourner NULL dans ce cas)

// - on stocke la position courante de str dans str_start
// avant de commencer à rechercher to_find dans str
// cela correspondra au pointeur potentiel vers le début de la sous-chaine trouvée dans str
// dans le cas où la sous-chaine est trouvée entièrement dans str, str_start sera retournée

// - on réinitialise to_find à to_find_start pour la comparaison
// (voir premier -) (même si la chaine n'a été trouvée que partiellement, la comparaison sera
// réinitialisée au premier caractère de to_find)
```

```
// pour recommencer la nouvelle comparaison au début de to_find

// - on compare les caractères de to_find et de str, deux à deux,
// en les incrémentant tant qu'ils sont égaux
// et que to_find n'a pas atteint la fin (dans ce cas, la sous-chaine a été trouvée)

// - on retourne str_start (le pointeur vers le début de la sous-chaine trouvée dans str)
// si la fin de to_find a été atteinte
// car cela signifie alors que toute la sous-chaine a été trouvée dans str

// - si to_find n'a pas été entièrement trouvée (pas du tout ou partiellement)
// on continue la recherche à partir du caractère suivant de str : str = str_start + 1

// - si on atteint la fin de str sans avoir trouvé to_find, on retourne NULL

// ATTENTION : inclure stddef.h pour pouvoir retourner NULL

#include <stddef.h>

char    *ft_strstr(char *str, char *to_find)
{
        char    *str_start;
        char    *to_find_start;

        // si to_find est une chaine vide
        if (*to_find == '\0')
                // retourner str
                return (str);
        // on initialise to_find_start à to_find
        to_find_start = to_find;
        // tant que la fin de str n'est pas atteinte
        while (*str != '\0')
        {
                // on réinitialise str_start à str
                str_start = str;
                // et to_find a to_find_start
                to_find = to_find_start;
                // tant que le caractère en cours de str est identique au caractère en cours de to_find
```

```
                // et que la fin de to_find n'est pas encore atteinte
                while (*str == *to_find && *to_find != '\0')
                {
                        // on passe au prochain caractère de str et de to_find
                        // pour effectuer une nouvelle comparaison
                        str++;
                        to_find++;
                }
                // si la fin de to_find a été atteinte
                // (si to_find a été entièrement trouvée dans str)
                if (*to_find == '\0')
                        // on retourne str_start
                        return (str_start);
                // si to_find n'a pas été entièrement trouvée (pas du tout ou partiellement)
                // on continue la recherche à partir du caractère suivant de str
                str = str_start + 1;
        }
        // si on atteint la fin de str sans avoir trouvé to_find, on retourne NULL
        return (NULL);
}

#include "ft_strcpy.h"
#include "ft_strstr.h"
#include "ft_putstr.h"

int     main(void)
{
        char    str[18];
        char    to_find[7];
        char    *res;

        ft_strcpy(str, "I need a needle !");
        ft_strcpy(to_find, "needle");
        res = ft_strstr(str, to_find);
        ft_putstr(res);
        return (0);
}
```

```
// Exemple :
// str = I need a needle !
// to_find = needle

// str          str_start       to_find_start   to_find
// I            I               n               n
// " "          " "             n               n
// n            n               n               n
// e            n               n               e
// e            n               n               e
// d            n               n               d
// " "          " "             n               n
// a            a               n               n
// n            n               n               n
// e            n               n               e
// e            n               n               e
// d            n               n               d
// l            n               n               l
// e            n               n               e
// \0           n               n               \0

// *to_find == '\0' => return (str_start)
// RESULTAT : needle !
```