ft_strjoin

Pour la compréhension de cette fonction, il est nécessaire pour vous d'avoir vu les fonctions suivantes :

- ft_strlen : Pour calculer la longueur d'une chaine de caractères.
- ft_strcat : Pour copier le contenu d'une chaine de caractères a la suite d'une autre.
- Et de connaitre un minimum la fonction malloc de la librairie 'stdlib.h'.

Bien, commençons.

ft_strjoin, ça sert à quoi?

ft_strjoin est une fonction qui relie deux chaines de caractères entre elle en y rajoutant un « séparateur » défini.

Cette fonction ne s'exécute pas seule et va avoir besoin de l'aide de d'autres fonctions qui sont les suivantes :

- ft_strcat
- ft_strlen
- ft_allstrlen : C'est une fonction créée pour l'occasion.

Concernant ft_allstrlen : C'est une fonction créée pour l'occasion, elle reprend la base de ft_strlen, voici son code :

```
int     ft_allstrlen(char **strs, int size)
{
        int i = 0;
        int result = 0;

        while(i < size) //tant que i est inferieur au nbrs de chaines dans strs
        {
        result = result + ft_strlen(strs[i]); //resulst += go fonction strlen qui va nous calculer la longueur des chaines
        i++; //on ajoute + 1 au 'i' pour defiler a la 2eme puis 3eme chaine de caracteres
        }
        return result; //return result quand on a parcouru tout notre tableau
}
```

Comment s'articule notre fonction ft_strjoin ?

1. Tout d'abord nous allons créer une variable de type 'int' qui va contenir la longueur de la taille de notre chaine de caractères finale (ici 'full_len'). On l'initialise a zero.

2. On test l'argument donne 'size', s'il vaut zero, on change la valeur de 'full_len' a 1, on va en avoir besoin pour malloc, ce serait inutile de faire une multiplication de zero.

3. Si size n'est pas égal a zero, on initialise un else, pour faire 3 actions :

- On appelle la fonction ft_allstrlen pour calculer la taille de toutes les chaines de caractères contenu dans **strs.
- On rajoute la taille de notre séparateur qu'on multiplie par size – 1 ; - 1 ici car on ne veut pas que séparateur soit placé a la toute fin.

- On rajoute une case a notre tableau pour lui laisser la place de mettre le signe NULL (‘\0’ en fin de phrase).

4. On crée une chaine de caractères (ici *strf) qui va recevoir toutes les chaines de caractères séparés par notre séparateur.

5. On utilise la fonction malloc pour donner à strf la taille totale (en prenant la valeur de full_len)

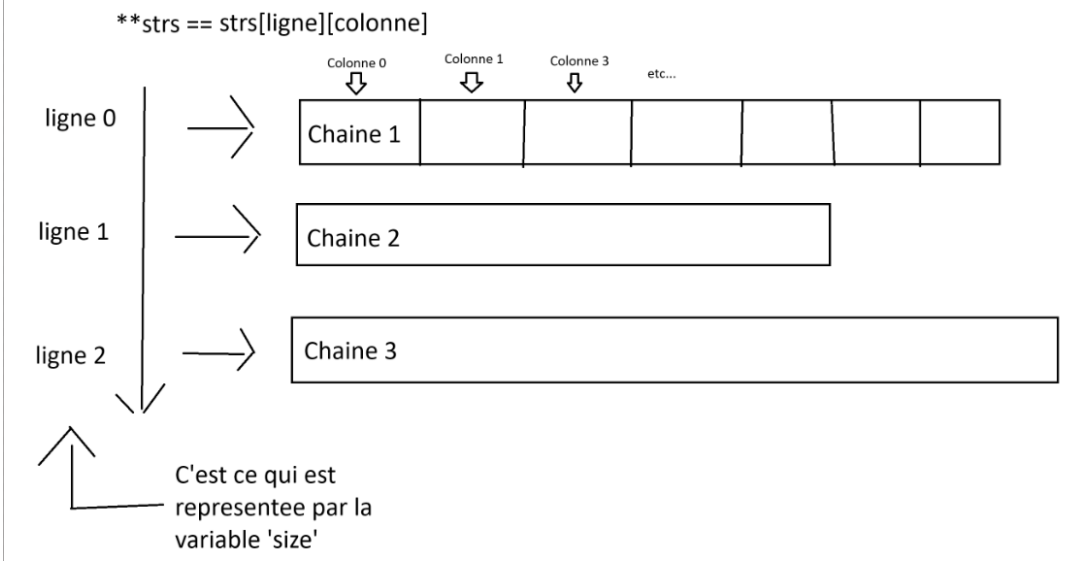
6. On initialise ‘i’ et on crée une boucle, la boucle prendra en paramètres -> tant que ‘i’ est inferieur a size (donc le nombre de phrases que contient **strs) alors :

    - On appelle strcat en lui donnant notre chaine de caractères finale, ainsi que strs[i], c’est-à-dire, la chaine de caractères vers laquelle strs pointe actuellement.
    - On crée une seconde boucle -> tant que i est différent de size -1 (Donc tant que ‘i’ n’a pas atteint la dernière ligne/dernière chaine de caractères) alors on envoie à str_cat, notre strf (notre chaine de caractères finale) ainsi que le séparateur. (Cette action va copier notre séparateur a la suite de la chaine de caractères copiée précédemment).
    - On peut sortir de notre boucle et évidemment on n’oublie pas d’incrémenter i++ ;
7. Pour finir, on retourne notre chaine de caractères finale.

Voici mon code de la fonction *ft_strjoin

```
char    *ft_strjoin(int size, char **strs, char *sep)
{
        int     full_len; //variable qui va contenir la taille de TOUTE la chaine finale
        full_len = 0;

        if(size == 0)
        {
                full_len = 1; //donne la possibilite d'utiliser malloc si size == 0
        }
        else
        {
                full_len = ft_allstrlen(strs, size); //On appelle ft_allstrlen pour calculer la taille de toutes les chaines de caracteres
                full_len += ft_strlen(sep) * (size - 1);//On rajoute la taille des separateurs (- 1 pour ne pas qu'un sep arrive a la toute fin de notre chaine de caracteres.
                full_len++; //on rajoute une case a full_len pour add notre '\0' a la fin
        }
        char *strf; //on cree la chaine de caracteres qui va tout recevoir

        strf = malloc(full_len); //on utilise malloc pour allouer la taille de full_len pour accueillir notre chaine

        int i;
        i = 0;

        while(i < size)
        {
                ft_strcat(strf, strs[i]); //on appelle strcat autant de fois qu'on a de chaine de caracteres differentes, on lui donne strf comme dest et strs[i] (donc les chaines) comme src
                if(i != size - 1)
                {
                        ft_strcat(strf, sep); //on appel strcat en lui donnant notre chaine de caracteres dest, a remplir par le sep
                }
                i++;
                // on incremente pour refaire une boucle et rajouter les autres chaines de caracteres.
        }
        return strf;
}
```

Petit rappel concernant pour lire correctement strs ->


**strs == strs[ligne][colonne]
Colonne 0
Colonne 1
Colonne 3
etc...
ligne 0
Chaine 1
ligne 1
Chaine 2
ligne 2
Chaine 3
C'est ce qui est
representee par la
variable 'size'