

```

// cette fonction convertit la chaine pointée par str en entier (de type int)
// tant que str suit les règles suivantes, elle retourne le nombre trouvé :
// - str peut commencer par un nombre arbitraire de white space
// (voir man isspace et man ascii) :
//     les white spaces correspondent à :
//     '\t' (horizontal tab) (Dec 9)
//     '\n' (new line) (Dec 10)
//     '\v' (vertical tab) (Dec 11)
//     '\f' (form feed) (Dec 12)
//     '\r' (carriage ret) (Dec 13)
//     SPACE (Dec 32)
// - str peut ensuite être suivi par un nombre arbitraire de signe + et de signe -
// le signe - fait changer le signe de l'entier retourné en fonction du nombre de signe -
// (si celui-ci est pair ou impair)
// - str doit, pour finir, être composé de chiffres de la base 10 (chiffres de 0 à 9)

// ft_atoi prend en paramètre la chaine de caractères pointée par str qu'elle doit convertir
// elle retourne un int (qui correspond à sa conversion)
int    ft_atoi(char *str)
{
    // pour déterminer si le nombre sera positif ou négatif
    int    sign;

    // pour stocker le résultat de la conversion en entier (int)
    // du caractère numérique en cours
    int    n;

    // pour stocker le résultat final de la conversion
    int    nbr;

    // on initialise sign à 1 (on suppose que le nombre est positif)
    // (voir plus bas))
    sign = 1;

```

```
// on initialise le résultat à 0
// pour commencer la conversion
nbr = 0;

// on ignore les white spaces :
// tant qu'on rencontre des white spaces
// (caractère de code ASCII Dec entre 9 et 13 (inclus) ou égal à 32)
while ((*str >= 9 && *str <= 13) || *str == 32)

    // on avance le pointeur d'un caractère
    // sans traitement (on les ignore)
    str++;

// on avance tant qu'on rencontre des - ou des +
// en comptant les signes - au fur et à mesure :
// tant que le caractère en cours est - ou +
while (*str == '-' || *str == '+')
{
    // si le caractère est -
    if (*str == '-')
        // on change le signe
        sign = -sign;
    // on avance dans la chaîne de caractères
    // (tant qu'on rencontre des - ou des +)
    str++;
}
```

```

// on lit chaque caractère numérique suivant
// tant que la chaîne de caractères n'a pas atteint la fin
// (chiffres en base 10, de 0 à 9)
while (*str != '\0' && *str >= '0' && *str <= '9')
{
    // pour chacun des chiffres
    // on le convertit en sa valeur entière
    // Exemple avec *str valant '2'
    // '2' équivaut à 50 en décimal (voir table ASCII)
    // '0' vaut 48 en décimal
    // l'opération se fait avec les valeurs décimales
    // des caractères :
    // 50 - 48 = 2
    // donc '2' - '0' donne bien 2
    // comme les valeurs décimales des chiffres en ASCII
    // (en caractères) se suivent
    // cette méthode fonctionne
    n = (*str - '0');

    // on met à jour le résultat
    // en décalant à gauche le résultat précédent
    // pour y ajouter à droite le chiffre en cours

    // si c'est le premier chiffre
    // nbr vaut 0
    // donc nbr = n
    // si ce n'est pas le premier chiffre
    // le résultat est multiplié par 10
    // cela à pour effet de décaler d'un chiffre
    // vers la gauche le chiffre existant
    // avant d'ajouter la nouvelle valeur numérique n

```

```

        // exemple :
        // 1er entier (1ère boucle) : 2
        // valeur précédente : 0
        // pas de décalage à gauche
        // (car  $0 * 10 = 0$ )
        // ajout de l'entier en cours :  $0 + 2 = 2$ 
        // 2ème entier (2ème boucle) : 9
        // valeur précédente : 2
        // décalage à gauche de 2 :
        // (car  $2 * 10 = 20$ )
        // ajout de l'entier en cours :  $20 + 9 = 29$ 

        nbr = nbr * 10 + n;

        // on passe au caractère suivant
        str++;
    }

    // on applique le signe déterminé (sign)
    // au résultat avant de retourner le tout

    return (sign * nbr);

}

```

```

#include "ft_strcpy.h"
#include "ft_atoi.h"
#include "ft_putnbr.h"

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

    ft_strcpy(str, "\t\n\v\f\r ++-+-12345abcd");
    res = ft_atoi(str);
}

```

```
    ft_putnbr(res);  
    write(1, "\n", 1);  
    return (0);
```

```
}
```

```
// RESULTAT : -12345
```