ft_atoi

ft_atoi est une fonction en C qui nous permet de convertir des caractères ASCII en Integer (int), le nom y fait d'ailleurs référence (a(scii)toi(nteger)).

Comme vous l'avez donc compris on va convertir des chaines de caractères ''1234'' en entier - > 1234.

Tout d'abord, jetons un œil à la table ASCII :

ASCII TABLE

Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	0	[NULL]	32	20	[SPACE]	64	40	@	96	60	`
1	1	[START OF HEADING]	33	21	!	65	41	A	97	61	a
2	2	[START OF TEXT]	34	22	"	66	42	B	98	62	b
3	3	[END OF TEXT]	35	23	#	67	43	C	99	63	c
4	4	[END OF TRANSMISSION]	36	24	$	68	44	D	100	64	d
5	5	[ENQUIRY]	37	25	%	69	45	E	101	65	e
6	6	[ACKNOWLEDGE]	38	26	&	70	46	F	102	66	f
7	7	[BELL]	39	27	'	71	47	G	103	67	g
8	8	[BACKSPACE]	40	28	(	72	48	H	104	68	h
9	9	[HORIZONTAL TAB]	41	29	)	73	49	I	105	69	i
10	A	[LINE FEED]	42	2A	*	74	4A	J	106	6A	j
11	B	[VERTICAL TAB]	43	2B	+	75	4B	K	107	6B	k
12	C	[FORM FEED]	44	2C	,	76	4C	L	108	6C	l
13	D	[CARRIAGE RETURN]	45	2D	-	77	4D	M	109	6D	m
14	E	[SHIFT OUT]	46	2E	.	78	4E	N	110	6E	n
15	F	[SHIFT IN]	47	2F	/	79	4F	O	111	6F	o
16	10	[DATA LINK ESCAPE]	48	30	0	80	50	P	112	70	p
17	11	[DEVICE CONTROL 1]	49	31	1	81	51	Q	113	71	q
18	12	[DEVICE CONTROL 2]	50	32	2	82	52	R	114	72	r
19	13	[DEVICE CONTROL 3]	51	33	3	83	53	S	115	73	s
20	14	[DEVICE CONTROL 4]	52	34	4	84	54	T	116	74	t
21	15	[NEGATIVE ACKNOWLEDGE]	53	35	5	85	55	U	117	75	u
22	16	[SYNCHRONOUS IDLE]	54	36	6	86	56	V	118	76	v
23	17	[END OF TRANS. BLOCK]	55	37	7	87	57	W	119	77	w
24	18	[CANCEL]	56	38	8	88	58	X	120	78	x
25	19	[END OF MEDIUM]	57	39	9	89	59	Y	121	79	y
26	1A	[SUBSTITUTE]	58	3A	:	90	5A	Z	122	7A	z
27	1B	[ESCAPE]	59	3B	;	91	5B	[	123	7B	{
28	1C	[FILE SEPARATOR]	60	3C	<	92	5C	\	124	7C	|
29	1D	[GROUP SEPARATOR]	61	3D	=	93	5D	]	125	7D	}
30	1E	[RECORD SEPARATOR]	62	3E	>	94	5E	^	126	7E	~
31	1F	[UNIT SEPARATOR]	63	3F	?	95	5F	_	127	7F	[DEL]

A gauche nous avons les « Decimal », qui sont des nombres, dans l'ordre croissant. C'est de cette façon que le tableau ASCII range ses caractères. A cote nous avons les « Hex » (*pour Hexadécimales*) qui est une base de 16 (les 10 premiers chiffres + les 6 premières lettres). Puis nous avons les « Char » qui sont des chaines de caractères.

Maintenant que vous avez ces notions, revenons à nos moutons, ce qui va nous intéresser ce sont les colonnes « Decimal » et « Char ». Ce qu'on cherche ici c'est de convertir une chaine de caractères (char) en décimal (les fameux integers).

Imaginons, nous avons une chaine de caractères « 12345 », ces caractères sont tous situes à la suite aux emplacements 49, 50, 51, 52, 53. Ce que l'on recherche à faire c'est de passer de ces décimales aux décimales des Integers (1, 2, 3, 4 et 5), pour cela dans notre code on va y intégrer cette ligne :

```
result = result * 10 + *str - '0';
```

Nous y reviendrons après, retenez juste que l'on soustrait le caractère '0' a un de nos caractères ci-dessus. Mais pourquoi cela ? tout simplement car on veut passer de notre valeur char à décimale, par exemple, notre '1' est situé à la valeur 49, ce que l'on veut c'est le faire passer à la valeur 1, pour cela, on soustrait le caractère '0' (ou valeur 48 en décimal).

Il faut vraiment voir le tableau Ascii comme un tableau qui se parcourt cases par cases.

Très bien, mais ft_atoi dans tout cela ?

Maintenant que cette notion a été vue, nous allons pouvoir passer au véritable sujet de ce document, la fonction atoi. Nous avons vu comment convertir un caractère en chiffres, mais ce ne sera pas tout ce qui sera demande, voici une liste non-exhaustive des choses qui seront également à gérer :

- Sauter les caractères non printable (Decimal 0 a 31 dans notre tableau ASCII)

- Sauter les espaces

- Prendre en compte le signe '–' et l'appliquer a notre conversion (pour rentre notre chiffre négatif ou au contraire le laisser en positif.)

Comment va se décomposer notre code ?

1. Pour commencer nous allons créer une variable de type int « sign », elle sera là pour stocker et vérifier si notre nombre est positif ou négatif. Nous l'initialiserons a 1.

2. Suivi d'une variable « result » également de type int qui sera la variable où nous stockerons nos éléments convertis.

   ---

3. On crée notre première boucle while : tant que notre chaine de caractères est soi un espace, un '\f', un '\n', un '\r', un '\t', ou un '\v', alors on incrémente notre chaine (C'est-à-dire, on passe au caractère suivant).

4. On crée notre deuxième boucle while : tant que notre chaine de caractères est un '-' on multiplie notre variable sign par sign – 1. Si vous vous rappelez de vos cours de maths, -n * -n ça fait +. Le nombre de '-' définira alors si nous devons sortir un nombre positif ou négatif. On incrémente notre chaine de caractères.

5. On crée notre troisième et dernière boucle while : Tant que notre chaine de caractères est supérieure ou égale a '0' et inferieur ou égale a '9', alors on fait la conversion. Vous vous rappelez de 'result = result * 10 + *str – '0' ;' ? Décomposons-le ! :

- Pour commencer nous multiplions le résultat par 10, pourquoi ? Car on ne veut pas remplacer les chiffres précédents ! Je m'explique, si nous convertissons un précèdent chiffre, par exemple 5, et que nous décidons d'ajouter 3 après, nous allons remplacer le 5 par la valeur des deux additionnés (5 + 3 = 8) Or, si nous multiplions 5 par 10 (donc 5 vaut 50, et qu'on lui ajoute 3 (donc, 53)) Nous aurons bien les chiffres à la suite des autres.
- Nous faisons la conversion du nombre suivant en lui enlevant '0' comme explique précédemment.
- Nous incrémentons évidemment notre chaine de caractères.

6. Pour finir, nous retournons notre variable « result » qui contient notre conversion, multiplie par notre « sign » qui rendra la valeur négative ou positive.

Mon propre code

cloe@DESKTOP-NKBHQGB: ~

```c
#include <unistd.h>
#include <stdio.h>

int     ft_atoi(char *str)
{
        int sign;
        int result = 0;
        sign = 1;
        while(*str == ' ' || *str == '\f' || *str == '\n' || *str == '\r' || *str == '\t' || *str == '\v')
        {
                str++;
        }
        while(*str == '-')
        {
                sign *= -1;
                str++;
        }
        while(*str >= '0' && *str <= '9')
        {
                result = result * 10 + *str - '0';
                str++;
        }

        return (result *= sign);

}


int main(void)
{
        char *str = "\n\t-53--6tr";
        printf("le resultat est : %d", ft_atoi(str));
        return 0;
}
~
~
```

Voici mon propre code pour la fonction ft_atoi, les étapes sont exactement les mêmes que les choses cités ci-dessus. L'ordre des boucles a une importance pour éviter de nous retrouver avec des caractères traites qui ne sont pas sensés l'être.
Voici le résultat une fois compilé :