

Capítulo III

Ejercicio 00 : ft_strlen


	Ejercicio: 00
	ft_strlen
	Directorio de entrega: <i>ex00/</i>
	Archivos a entregar: ft_strlen.c
	Funciones autorizadas: Ninguna

- Escribe una función que cuente el número de caracteres de un string y que devuelva el número encontrado.
- El prototipo de la función deberá ser el siguiente:

```
int ft_strlen(char *str);
```

Capítulo IV

Ejercicio 01 : ft_putstr


	Ejercicio: 01
	ft_putstr
	Directorio de entrega: <i>ex01/</i>
	Archivos a entregar: ft_putstr.c
	Funciones autorizadas: write

- Escribe una función que muestre uno a uno en la pantalla los caracteres de una cadena de caracteres.
- La dirección del primer carácter de la cadena de caracteres está incluida en el puntero usado como parámetro a la función.
- El prototipo de la función deberá ser el siguiente:

```
void    ft_putstr(char *str);
```

Capítulo V

Ejercicio 02 : ft_putnbr

	Ejercicio: 02
	ft_putnbr
	Directorio de entrega: <i>ex02/</i>
	Archivos a entregar: ft_putnbr.c
	Funciones autorizadas: write


- Escribe una función que muestre un número pasado como parámetro. La función deberá ser capaz de mostrar todos los valores posibles en una variable de tipo `int`.
- El prototipo de la función deberá ser el siguiente:

```
void ft_putnbr(int nb);
```

- Por ejemplo:
 - `ft_putnbr(42)` muestra “42”.

Capítulo VI

Ejercicio 03 : ft_atoi

	Ejercicio: 03
	ft_atoi
	Directorio de entrega: <i>ex03/</i>
	Archivos a entregar: ft_atoi.c
	Funciones autorizadas: Ninguna

- Escribe una función que convierta el principio del string apuntado por str en un entero de tipo int
- str puede empezar con un número arbitrario de espacios (tal y como lo define `isspace(3)`)
- str puede ir seguido de un número arbitrario de signos + y de signos -. El signo - hará cambiar el signo del entero devuelto en función del número de signos - y si este es par o impar.
- str puede ir seguido de cualquier cantidad de número de dígitos en base 10
- Tu función tendrá que leer los caracteres de str, siempre que estos cumplan con las reglas mencionadas anteriormente, y tendrá que devolver el número encontrado hasta entonces.
- No deberías tener en cuenta los desbordamientos (overflows y underflows), en estos casos el resultado se considera indefinido.
- Puedes comparar tu función con la verdadera función atoi, quitando la parte de los signos y del overflow.
- Aquí tienes el ejemplo de un programa que muestra el valor devuelto por atoi:

```
$>./a.out " ----++1234ab567"  
-1234
```

- El prototipo de la función deberá ser el siguiente:

```
int ft_atoi(char *str);
```