LIBFT

PARTE 1 – Funciones de Libc.

Francisco Rubio Illán

frubio-i
frubio-i@student.42barcelona.com

ft_isalpha

Prototipo:

```
int ft_isalpha(int c);
```

Parámetros:

c: Un carácter cuyo tipo de letra se desea verificar.

Valor devuelto:

1 si el carácter es una letra alfabética.

0 en caso contrario.

Funciones autorizadas:

Ninguna.

Descripción:

Esta función verifica si un carácter dado es una letra alfabética. Retorna 1 si el carácter está entre 'A' y 'Z' o entre 'a' y 'z', indicando así que es una letra alfabética. En todos los demás casos, retorna 0, lo que indica que el carácter no es una letra alfabética. Esta función es útil para validar entradas de texto en programas que requieren especificaciones de formato estrictas, especialmente cuando se trabaja con cadenas de caracteres que pueden contener una mezcla de letras y otros caracteres no alfabéticos.

ft_isdigit

Prototipo:

int ft_isdigit(int c);

Parámetros:

c: Un carácter cuyo tipo numérico se desea verificar.

Valor devuelto:

1 si el carácter es un dígito numérico (entre '0' y '9').

0 en caso contrario.

Funciones autorizadas:

Ninguna.

Descripción:

Esta función determina si un carácter dado es un dígito numérico. Retorna 1 si el carácter está entre '0' y '9', lo que indica que es un dígito numérico. Para cualquier otro carácter, retorna 0, señalando que no es un dígito numérico. Es particularmente útil en operaciones que involucran la validación de entradas de texto, permitiendo filtrar o procesar solo aquellos caracteres que representan números, lo cual es crucial en aplicaciones que manejan datos numéricos, como calculadoras, formularios web, y sistemas de registro de inventario.

ft_isalnum

Prototipo:

int ft_isalnum(int c);

Parámetros:

c: Un carácter cuyo tipo alfanumérico se desea verificar.

Valor devuelto:

1 si el carácter es alfanumérico (letra o dígito).

0 en caso contrario.

Funciones autorizadas:

Ninguna.

Descripción:

Esta función verifica si un carácter dado es alfanumérico, es decir, si es una letra (alfabética) o un dígito numérico. Retorna 1 si el carácter está entre 'A' y 'Z' o entre 'a' y 'z' (letras alfabéticas) o entre '0' y '9' (dígitos numéricos). Para cualquier otro carácter, retorna 0, indicando que no es alfanumérico. Esta función es útil en la validación de entradas de texto en programas que requieren especificaciones de formato estrictas, especialmente cuando se trabaja con cadenas de caracteres que pueden contener una mezcla de letras, dígitos y otros caracteres no alfanuméricos.

ft_ascii

Prototipo:

int ft_isascii(int c);

Parámetros:

c: Un carácter cuyo valor ASCII se desea verificar

Valor devuelto:

1 si el carácter tiene un valor ASCII imprimible.

0 en caso contrario.

Funciones autorizadas:

Ninguna.

Descripción:

Esta función verifica si un carácter dado tiene un valor ASCII imprimible, es decir, si su valor ASCII está dentro del rango de caracteres imprimibles definidos por el estándar ASCII. Retorna 1 si el valor ASCII del carácter está entre 0 y 127, lo que indica que es un carácter imprimible. Para cualquier otro valor, retorna 0, señalando que el carácter no es imprimible. Esta función es útil para validar entradas de texto en programas que deben manejar caracteres dentro del rango ASCII estándar, asegurando que solo se procesen caracteres válidos y evitando errores o comportamientos inesperados con caracteres no imprimibles.

ft_isprint

Prototipo:

Int ft_isprint(int c);

Parámetros:

c: Un carácter cuyo valor imprimible se desea verificar.

Valor devuelto:

1 si el carácter es imprimible según el estándar ASCII.

0 en caso contrario.

Funciones autorizadas:

Ninguna.

Descripción:

Esta función verifica si un carácter dado es imprimible según el estándar ASCII, es decir, si su valor ASCII corresponde a un carácter que puede ser impreso en una pantalla o dispositivo de salida. Retorna 1 si el valor ASCII del carácter está entre 32 (' ') y 126 ('~'), lo que indica que es un carácter imprimible. Para cualquier otro valor, incluidos los caracteres control (como tabulación, nueva línea, retorno de carro, etc.) y los caracteres fuera del rango imprimible, retorna 0, señalando que el carácter no es imprimible. Esta función es útil para validar entradas de texto en programas que deben manejar solo caracteres legibles por humanos, ayudando a evitar la inclusión de caracteres especiales o control que podrían causar problemas en la visualización o procesamiento de textos.

ft_strlen

Prototipo:

size_t ft_strlen(const char *s);

Parámetros:

s: Una cadena de caracteres (const char *) cuya longitud se desea conocer.

Valor devuelto:

La longitud de la cadena s, excluyendo el carácter nulo final.

Funciones autorizadas:

Ninguna.

Descripción:

Esta función calcula y devuelve la longitud de una cadena de caracteres, es decir, el número de caracteres antes del terminador nulo (\0). No cuenta el carácter nulo final. Es fundamental para muchas operaciones de manipulación de cadenas, ya que permite saber cuántos elementos contiene una cadena sin tener que recorrerla hasta encontrar el carácter nulo.

ft_memset

Prototipo:

```
void *ft_memset(void *s, int c, size_t n);
```

Parámetros:

s: Un puntero a la ubicación inicial donde comenzará la operación de relleno.

c: El carácter (o byte) con el que se llenará la región de memoria.

n: El número de bytes a rellenar, comenzando desde la ubicación apuntada por s.

Valor devuelto:

Un puntero al bloque de memoria original s.

Funciones autorizadas:

Ninguna.

Descripción:

Esta función rellena la primera n bytes de la región de memoria apuntada por s con el carácter c. Es equivalente a copiar n veces el carácter c a la dirección base s. Esta operación es comúnmente utilizada para inicializar bloques de memoria a un valor específico, como el carácter nulo ('\0') para terminar cadenas de caracteres o para limpiar áreas de memoria previamente asignadas.

ft_bzero

Prototipo:

```
void ft_bzero(void *s, size_t n);
```

Parámetros:

s: Un puntero a la ubicación inicial donde comenzará la operación de relleno.

n: El número de bytes a rellenar, comenzando desde la ubicación apuntada por s.

Valor devuelto:

Ninguno.

Funciones autorizadas:

Ninguno.

Descripción:

Esta función rellena la primera n bytes de la región de memoria apuntada por s con el carácter nulo ('\0'). Específicamente, establece cada uno de los primeros n bytes de la memoria apuntada por s a 0. Esta operación es útil para inicializar bloques de memoria a un estado seguro, asegurando que no contengan datos residuales o indeseables antes de su uso.

ft_memcpy

Prototipo:

```
void *ft_memcpy(void *dest, const void *src, size_t n);
```

Parámetros:

dest: Un puntero a la ubicación donde se copiarán los datos.

src: Un puntero a la fuente de los datos que se van a copiar.

n: El número de bytes a copiar desde src a dest.

Valor devuelto:

Un puntero al bloque de memoria dest.

Funciones autorizadas:

Ninguna.

Descripción:

Esta función copia n bytes de la memoria apuntada por src a la memoria apuntada por dest. Es importante notar que dest y src pueden ser la misma ubicación en memoria; en tal caso, ft_memcpy simplemente sobrescribe los datos existentes. Esta función es útil para copiar bloques de memoria de manera eficiente, especialmente cuando se necesita duplicar datos o moverlos dentro de la memoria.

ft_memmove

Prototipo:

```
void *ft_memmove(void *dest, const void *src, size_t n);
```

Parámetros:

dest: Un puntero a la ubicación donde se copiarán los datos.

src: Un puntero a la fuente de los datos que se van a copiar.

n: El número de bytes a copiar desde src a dest.

Valor devuelto:

Un puntero al bloque de memoria dest.

Funciones autorizadas:

Ninguna.

Descripción:

Esta función copia n bytes de la memoria apuntada por src a la memoria apuntada por dest. A diferencia de ft_memcpy, ft_memmove garantiza que la operación se realice correctamente incluso si la zona de destino y la zona de origen se superponen, es decir, si dest y src apuntan a la misma ubicación en memoria o si dest está dentro del rango de src. Esto hace que ft_memmove sea más seguro usar cuando se sabe que las zonas de memoria pueden solaparse.

ft_strlcpy

Prototipo:

```
size_t    ft_strlcpy(char *dst, const char *src, size_t size);
```

Parámetros:

dst: Un puntero a la cadena de destino donde se copiarán los datos.

src: Un puntero a la cadena fuente de los datos que se van a copiar.

size: El tamaño total disponible para la cadena de destino, incluido el carácter nulo final.

Valor devuelto:

El tamaño total de la cadena fuente src, incluido el carácter nulo final.

Funciones autorizadas:

Ninguno.

Descripción:

Esta función copia la cadena src a dst, asegurándose de que dst esté terminada con un carácter nulo (\0) y que no se escriban más de size - 1 caracteres en dst. Si size es igual al tamaño de src, entonces dst será exactamente igual a src, pero aún así se añadirá el carácter nulo final a dst. Esta función es útil para copiar cadenas de manera segura, evitando desbordamientos de búfer al limitar el número de caracteres copiados a size - 1.

ft_strlcat

Prototipo:

```
size_t    ft_strlcat(char *dst, const char *src, size_t size);
```

Parámetros:

dst: Un puntero a la cadena de destino donde se añadirán los datos.

src: Un puntero a la cadena fuente de los datos que se van a añadir.

size: El tamaño total disponible para la cadena de destino, incluido el espacio para el carácter nulo final.

Valor devuelto:

El tamaño total de la cadena resultante, incluido el carácter nulo final.

Funciones autorizadas:

Ninguna.

Descripción:

Esta función concatena la cadena src al final de dst, asegurándose de que dst esté terminada con un carácter nulo (\0) y que no se escriban más de size - 1 caracteres en dst. Si size es menor que el tamaño de src, entonces solo se copian size - strlen(dst) - 1 caracteres de src a dst, y se añade el carácter nulo final. Esta función es útil para concatenar cadenas de manera segura, evitando desbordamientos de búfer al limitar el número de caracteres añadidos a size - 1.

ft_toupper

Prototipo:

```
int  ft_toupper(int c);
```

Parámetros:

c: Un carácter cuyo valor ASCII se desea convertir a mayúsculas.

Valor devuelto:

El carácter c convertido a mayúsculas si es posible, o el carácter original si no es convertible a mayúsculas.

Funciones autorizadas:

Ninguna.

Descripción:

Esta función convierte un carácter dado c a su correspondiente versión en mayúsculas, según el estándar ASCII. Retorna el carácter c convertido a mayúsculas si es posible (es decir, si c es una letra minúscula), o el carácter original si no es convertible a mayúsculas (por ejemplo, espacios, signos de puntuación, etc.). Esta función es útil para normalizar el caso de caracteres individuales en cadenas de texto, asegurando que todas las versiones de un carácter sean consistentemente tratadas como iguales en términos de comparación de cadenas.

ft_tolower

Prototipo:

```
int  ft_tolower(int c);
```

Parámetros:

c: Un carácter cuyo valor ASCII se desea convertir a minúsculas.

Valor devuelto:

El carácter c convertido a minúsculas si es posible, o el carácter original si no es convertible a minúsculas.

Funciones autorizadas:

Ninguna.

Descripción:

Esta función convierte un carácter dado c a su correspondiente versión en minúsculas, según el estándar ASCII. Retorna el carácter c convertido a minúsculas si es posible (es decir, si c es una letra mayúscula), o el carácter original si no es convertible a minúsculas (por ejemplo, espacios, signos de puntuación, etc.). Esta función es útil para normalizar el caso de caracteres individuales en cadenas de texto, asegurando que todas las versiones de un carácter sean consistentemente tratadas como iguales en términos de comparación de cadenas.

ft_strchr

Prototipo:

```
char *ft_strchr(const char *s, int c);
```

Parámetros:

s: Una cadena de caracteres (const char *) en la que buscar el carácter c.

c: El carácter que se busca en la cadena s.

Valor devuelto:

Un puntero al primer carácter encontrado en s que coincide con c, o NULL si no se encuentra ningún carácter coincidente.

Funciones autorizadas:

Ninguna.

Descripción:

Esta función busca el primer carácter c en la cadena s. Retorna un puntero al primer carácter encontrado que coincide con c, o NULL si no se encuentra ningún carácter coincidente. Esta función es útil para localizar la posición de un carácter específico dentro de una cadena, lo cual es común en diversas operaciones de manipulación de cadenas, como sustituciones condicionales o extracciones basadas en patrones.

ft_strrchr

Prototipo:

char *ft_strrchr(const char *s, int c);

Parámetros:

s: Una cadena de caracteres (const char *) en la que buscar el último carácter c.

c: El carácter que se busca en la cadena s.

Valor devuelto:

Un puntero al último carácter encontrado en s que coincide con c, o NULL si no se encuentra ningún carácter coincidente.

Funciones autorizadas:

Ninguna.

Descripción:

Esta función busca el último carácter c en la cadena s. Retorna un puntero al último carácter encontrado que coincide con c, o NULL si no se encuentra ningún carácter coincidente. Diferenciándose de ft_strchr, ft_strrchr comienza la búsqueda desde el final de la cadena hacia el principio, lo que la hace ideal para situaciones donde se necesita encontrar la última ocurrencia de un carácter específico en una cadena.

ft_strncmp

Prototipo:

```
int  ft_strncmp(const char *s1, const char *s2, size_t n);
```

Parámetros:

s1: Una cadena de caracteres (const char *) que se compara con s2.

s2: Otra cadena de caracteres (const char *) que se compara con s1.

n: El número máximo de caracteres a comparar entre s1 y s2.

Valor devuelto:

Un entero que indica el resultado de la comparación:

- Menor que 0 si s1 es lexicográficamente anterior a s2.
- Cero si s1 y s2 son iguales hasta el n-ésimo carácter.
- Mayor que 0 si s1 es lexicográficamente posterior a s2.

Funciones autorizadas:

Ninguna.

Descripción:

Esta función compara las primeras n posiciones de las cadenas s1 y s2. Retorna un entero que indica el resultado de la comparación lexicográfica de estas dos cadenas. Es útil para comparar cadenas de manera flexible, permitiendo especificar un límite en la cantidad de caracteres a comparar, lo cual es especialmente útil en situaciones donde las cadenas pueden tener diferentes longitudes o cuando se quiere ignorar el resto de la cadena después de cierto punto.

ft_memchr

Prototipo:

```
void *ft_memchr(const void *s, int c, size_t n);
```

Parámetros:

s: Un puntero a la ubicación inicial donde comenzará la búsqueda.

c: El valor del carácter que se busca.

n: El número de bytes a buscar en la memoria apuntada por s.

Valor devuelto:

Un puntero al primer byte encontrado que coincide con c, o NULL si no se encuentra ninguna coincidencia.

Funciones autorizadas:

Ninguna.

Descripción:

Esta función busca el primer byte en la memoria apuntada por s que coincide con el valor del carácter c. Retorna un puntero al primer byte encontrado que coincide con c, o NULL si no se encuentra ninguna coincidencia. Esta función es útil para localizar rápidamente una secuencia de bytes específicos dentro de un bloque de memoria, lo cual es común en varias operaciones de manipulación de datos binarios, como la búsqueda de marcadores o delimitadores dentro de estructuras de datos.

ft_strnstr

Prototipo:

```
char *ft_strnstr(const char *big, const char *little, size_t len);
```

Parámetros:

big: Una cadena de caracteres (const char *) en la que buscar la subcadena little.

little: Una cadena de caracteres (const char *) que se busca dentro de big.

len: El número máximo de caracteres a considerar dentro de big durante la búsqueda.

Valor devuelto:

Un puntero al inicio de la primera aparición de little dentro de big, o NULL si little no se encuentra dentro de big o si little es una cadena vacía.

Funciones autorizadas:

Ninguna.

Descripción:

Esta función busca la primera aparición de la subcadena little dentro de la cadena big, considerando solo los primeros len caracteres de big. Retorna un puntero al inicio de la primera aparición encontrada de little dentro de big, o NULL si little no se encuentra dentro de big o si little es una cadena vacía. Esta función es útil para realizar búsquedas de subcadenas dentro de cadenas de manera eficiente, limitando el alcance de la búsqueda a un segmento específico de la cadena principal.

ft_atoi

Prototipo:

```
int  ft_atoi(const char *nptr);
```

Parámetros:

nptr: Un puntero a una cadena de caracteres que representa un número entero en forma de cadena.

Valor devuelto:

El valor entero correspondiente a la cadena apuntada por nptr.

Funciones autorizadas:

Ninguna.

Descripción:

Esta función convierte una cadena de caracteres que representa un número entero en su valor numérico correspondiente. Comienza analizando los caracteres de la cadena desde el principio hasta que encuentra un carácter que no es parte del conjunto numérico (caracteres no numéricos, signos de puntuación, etc.), o hasta que alcanza el final de la cadena. Los valores numéricos pueden ser positivos o negativos, y la función ignora los caracteres de signo '+' y '-' al principio de la cadena, excepto el primero si está presente. Retorna el valor entero correspondiente a la cadena apuntada por nptr.

ft_calloc

Prototipo:

```
void *ft_calloc(size_t count, size_t size);
```

Parámetros:

count: El número de elementos a reservar.

size: El tamaño de cada elemento a reservar.

Valor devuelto:

Un puntero al bloque de memoria reservado, o NULL si la reserva falla.

Funciones autorizadas:

Ninguna.

Descripción:

Esta función reserva un bloque de memoria de tamaño suficiente para almacenar count elementos de size bytes cada uno, inicializando todos los bytes reservados a cero. Es similar a malloc, pero con la adición de la inicialización de los bytes reservados a cero. Esto es particularmente útil cuando se necesita trabajar con arrays de números enteros o flotantes, ya que garantiza que todos los elementos estén inicializados a un valor conocido y seguro, evitando comportamientos indefinidos o datos residuales de otras operaciones.

ft_strdup

Prototipo:

```
char *ft_strdup(const char *s1);
```

Parámetros:

s1: Una cadena de caracteres (const char *) cuya copia se desea crear.

Valor devuelto:

Un puntero a la nueva cadena creada, que es una copia de s1. Retorna NULL si la operación falla.

Funciones autorizadas:

Ninguna.

Descripción:

Esta función crea y devuelve una copia de la cadena s1. La nueva cadena tiene el mismo contenido que s1 y debe ser liberada manualmente utilizando free cuando ya no sea necesaria. Es útil para duplicar cadenas de caracteres, especialmente cuando se necesita modificar una copia de una cadena sin afectar la original.

PARTE 2 – Funciones adicionales.

ft_substr

Prototipo:

char *ft_substr(char const *s, unsigned int start, size_t len);

Parámetros:

s: La string desde la que crear la substring.

start: El índice del caracter en ’s’ desde el que empezar la substring.

len: La longitud máxima de la substring.

Valor devuelto:

La substring resultante.

NULL si falla la reserva de memoria.

Funciones autorizadas:

Malloc.

Descripción:

Reserva (con malloc(3)) y devuelve una substring de la string ’s’.

La substring empieza desde el índice ’start’ y tiene una longitud máxima ’len’.

ft_strjoin

Prototipo:

char *ft_strjoin(char const *s1, char const *s2);

Parámetros:

s1: La primera string.

s2: La string a añadir a ’s1’.

Valor devuelto:

La nueva string.

NULL si falla la reserva de memoria.

Funciones autorizadas:

Malloc.

Descripción:

Reserva (con malloc(3)) y devuelve una nueva string, formada por la concatenación de ’s1’ y ’s2’.

ft_strtrim

Prototipo:

char *ft_strtrim(char const *s1, char const *set);

Parámetros:

s1: La string que debe ser recortada.

set: Los caracteres a eliminar de la string.

Valor devuelto:

La string recortada.

NULL si falla la reserva de memoria.

Funciones autorizadas:

Malloc.

Descripción:

Elimina todos los caracteres de la string ’set’ desde el principio y desde el final de ’s1’, hasta encontrar un caracter no perteneciente a ’set’. La string resultante se devuelve con una reserva de malloc(3).

ft_split

Prototipo:

char **ft_split(char const *s, char c);

Parámetros:

s: La string a separar.

c: El carácter delimitador.

Valor devuelto:

El array de nuevas strings resultante de la separación.

NULL si falla la reserva de memoria.

Funciones autorizadas:

Malloc, free.

Descripción:

Reserva (utilizando malloc(3)) un array de strings resultante de separar la string ’s’ en substrings utilizando el caracter ’c’ como delimitador. El array debe terminar con un puntero NULL.

ft_itoa

Prototipo:

char *ft_itoa(int n);

Parámetros:

n: el entero a convertir.

Valor devuelto:

La string que represente el número.

NULL si falla la reserva de memoria.

Funciones autorizadas:

Malloc.

Descripción:

Utilizando malloc(3), genera una string que represente el valor entero recibido como argumento. Los números negativos tienen que gestionarse.

ft_strmapi

Prototipo:

char *ft_strmapi(char const *s, char (*f)(unsigned int, char));

Parámetros:

s: La string que iterar.

f: La función a aplicar sobre cada carácter.

Valor devuelto:

La string creada tras el correcto uso de ’f’ sobre

cada carácter.

NULL si falla la reserva de memoria.

Funciones autorizadas:

Malloc.

Descripción:

Aplica la función ’f’ a cada carácter de la cadena ’s’, pasando su índice como primer argumento y el propio carácter como segundo argumento. Se crea una nueva cadena (utilizando malloc(3)) para recoger los resultados de las sucesivas aplicaciones de ’f’.

ft_striteri

Prototipo:

```
void ft_striteri(char *s, void (*f)(unsigned int, char*));
```

Parámetros:

s: La string que iterar.

f: La función a aplicar sobre cada carácter.

Valor devuelto:

Ninguno.

Funciones autorizadas:

Ninguna.

Descripción:

A cada carácter de la string ’s’, aplica la función ’f’ dando como parámetros el índice de cada carácter dentro de ’s’ y la dirección del propio carácter, que podrá modificarse si es necesario.

ft_putchar_fd

Prototipo:

```
void ft_putchar_fd(char c, int fd);
```

Parámetros:

c: El carácter a enviar.

fd: El file descriptor sobre el que escribir.

Valor devuelto:

Ninguno.

Funciones autorizadas:

Write.

Descripción:

Envía el carácter ’c’ al file descriptor especificado.

ft_putstr_fd

Prototipo:

```
void ft_putstr_fd(char *s, int fd);
```

Parámetros:

s: La string a enviar.

fd: El file descriptor sobre el que escribir.

Valor devuelto:

Ninguno.

Funciones autorizadas:

Write.

Descripción:

Envía la string ’s’ al file descriptor especificado.

ft_putendl_fd

Prototipo:

```
void ft_putendl_fd(char *s, int fd);
```

Parámetros:

s: La string a enviar.

fd: El file descriptor sobre el que escribir.

Valor devuelto:

Ninguno.

Funciones autorizadas:

Write.

Descripción:

Envía la string ’s’ al file descriptor dado, seguido de un salto de línea.

ft_putnbr_fd

Prototipo:

void ft_putnbr_fd(int n, int fd);

Parámetros:

n: El número que enviar.

fd: El file descriptor sobre el que escribir.

Valor devuelto:

Ninguno.

Funciones autorizadas:

Write.

Descripción:

Envía el número ’n’ al file descriptor dado.

ft_FORMATO

Prototipo:

Parámetros:

Valor devuelto:

Funciones autorizadas:

Descripción: