libre

Tu primera biblioteca propia

Resumen:
este proyecto trata sobre codificar una biblioteca C.
Contendrá muchas funciones de propósito general en las que dependerán sus programas.

Versión: 15

Contenido

Capítulo I

Introducción

La programación en C puede resultar muy tediosa cuando no se tiene acceso a las funciones estándar de gran utilidad. Este proyecto trata de comprender la forma en que funcionan estas funciones, implementarlas y aprender a usarlas. Crearás tu propia biblioteca. Será útil ya que lo utilizará en sus próximas tareas escolares C.

Tómese el tiempo para ampliar su libertad durante todo el año. Sin embargo, cuando trabaje en un proyecto nuevo, no olvide asegurarse de que las funciones utilizadas en su biblioteca estén permitidas en las pautas del proyecto.

Capitulo dos

Instrucciones comunes

- Su proyecto debe estar escrito en C.
- Su proyecto debe estar escrito de acuerdo con la Norma. Si tiene archivos/funciones adicionales, se incluyen en la verificación de normas y recibirá un 0 si hay un error de norma dentro.
- Sus funciones no deberían cerrarse inesperadamente (fallo de segmentación, error de bus, doble liberación, etc.) aparte de comportamientos indefinidos. Si esto sucede, su proyecto se considerará no funcional y recibirá un 0 durante la evaluación.
- Todo el espacio de memoria asignado al montón debe liberarse adecuadamente cuando sea necesario. Sin fugas será tolerado.
- Si el tema lo requiere, debe enviar un Makefile que compilará sus archivos fuente en la salida requerida con los indicadores -Wall, -Wextra y -Werror, use cc y su Makefile no debe volver a vincularse.
- Su Makefile debe contener al menos las reglas $(NAME), all, clean, fclean y re.
- Para entregar bonos a su proyecto, debe incluir un bono de regla en su Makefile, que agregará todos los encabezados, bibliotecas o funciones que están prohibidas en la parte principal del proyecto. Los bonos deben estar en un archivo diferente _bonus.{c/h} si el asunto no especifica nada más. La evaluación de las partes obligatoria y de bonificación se realiza por separado.
- Si su proyecto le permite usar su libft, debe copiar sus fuentes y su Makefile asociado en una carpeta libft con su Makefile asociado. El Makefile de su proyecto debe compilar la biblioteca usando su Makefile y luego compilar el proyecto.
- Le recomendamos que cree programas de prueba para su proyecto, aunque este trabajo no tendrá que enviarse ni será calificado. Le dará la oportunidad de probar fácilmente su trabajo y el de sus compañeros. Estas pruebas le resultarán especialmente útiles durante su defensa. En efecto, durante la defensa, usted es libre de utilizar sus pruebas y/o las pruebas del compañero que está evaluando.
- Envíe su trabajo a su repositorio git asignado. Sólo se calificará el trabajo en el repositorio de git. Si se asigna a Deepthinkt para calificar su trabajo, se hará

después de sus evaluaciones de pares. Si ocurre un error en cualquier sección de su trabajo durante la calificación de Deepthink, la evaluación se detendrá.

Capítulo III

Parte obligatoria

Nombre del programa	libft.a
Entregar archivos	Archivo Make, libft.h, ft_*.c
Funciones	NOMBRE, todo, limpiar, fclean, re
externas de Makefile.	Detallado abajo
Libft autorizado	n / A
Descripción	Escribe tu propia biblioteca: una colección de funciones. Esa será una herramienta útil para tu curso.

III.1 Consideraciones técnicas

- Está prohibido declarar variables globales.
- Si necesita funciones auxiliares para dividir una función más compleja, defínalas como estáticas funciones. De esta forma, su alcance se limitará al fichero correspondiente.
- Coloque todos sus archivos en la raíz de su repositorio.
- Está prohibido entregar archivos no utilizados.
- Cada archivo .c debe compilarse con las banderas -Wall -Wextra -Werror.
- Debes usar el comando ar para crear tu biblioteca. Usando el comando libtool está prohibido.
- Su libft.a debe crearse en la raíz de su repositorio.

III.2 Parte 1 - Funciones Libc

Para comenzar, debes rehacer un conjunto de funciones de la libc. Tus funciones tendrán los mismos prototipos e implementarán los mismos comportamientos que las originales. Deben cumplir con la forma en que están definidos en su hombre. La única diferencia serán sus nombres. Comenzarán con el prefijo 'ft_'. Por ejemplo, strlen se convierte en ft_strlen.

Algunos de los prototipos de funciones que debe rehacer utilizan el calificador 'restringir'. Esta palabra clave es parte del estándar c99. Por lo tanto, está prohibido incluirlo en sus propios prototipos y compilar su código con el indicador -std=c99.

Debes escribir tu propia función implementando las siguientes originales. No requieren ninguna función externa:

- isalfa
- es dígito
- isalno
- isascii
- impresión
- esforzarse
- conjunto de memorias
- bcero
- memoria
- memmover
- strlcpy
- strlcat
- adorno
- reducir
- strchr
- strrchr
- strncmp
- memchr
- memcmp
- strnstr
- atoí

Para implementar las dos funciones siguientes, utilizará malloc():

- calloc
- strdup

III.3 Parte 2 - Funciones adicionales

En esta segunda parte, debes desarrollar un conjunto de funciones que no están en la libc, o que forman parte de él pero de forma diferente.

Algunas de las siguientes funciones pueden ser útiles para escribir el funciones de la Parte 1.

Nombre de la función	ft_substr
Prototipo	char *ft_substr(char const *s, inicio int sin signo, tamaño_t len);
Entregar archivos	-
Parámetros	s: La cadena a partir de la cual crear la subcadena. inicio: el índice inicial de la subcadena en el cadena 's'. len: la longitud máxima de la subcadena.
Valor de retorno	La subcadena. NULL si la asignación falla.
Funciones externas.	malloc
Descripción	Asigna (con malloc(3)) y devuelve una subcadena de la cadena 's'. La subcadena comienza en el índice 'inicio' y es de tamaño máximo 'len'.

Nombre de la función	ft_strjoin
Prototipo	char *ft_strjoin(char const *s1, char const *s2);
Entregar archivos	-
Parámetros	s1: la cadena de prefijo. s2: la cadena de sufijo.
Valor de retorno	La nueva cuerda. NULL si la asignación falla.
Funciones externas.	malloc
Descripción	Asigna (con malloc(3)) y devuelve un nuevo cadena, que es el resultado de la concatenación de 's1' y 's2'.

Nombre de la función	ft_strtrim
Prototipo	char *ft_strtrim(char const *s1, char const *set);
Entregar archivos	-
Parámetros	s1: La cuerda que se va a recortar. set: el conjunto de caracteres de referencia para recortar.
Valor de retorno	La cuerda recortada. NULL si la asignación falla.
Funciones externas.	malloc
Descripción	Asigna (con malloc(3)) y devuelve una copia de 's1' con los caracteres especificados en 'set' eliminados desde el principio y el final de la cadena.

Nombre de la función	ft_split
Prototipo	char **ft_split(char const *s, char c);
Entregar archivos	-
Parámetros	s: la cadena que se va a dividir. c: El carácter delimitador.
Valor de retorno	La matriz de nuevas cadenas resultantes de la división. NULL si la asignación falla.
Funciones externas.	malloc, gratis
Descripción	Asigna (con malloc(3)) y devuelve una matriz de cadenas obtenidas dividiendo 's' usando el carácter 'c' como delimitador. La matriz debe terminar con un puntero NULO.

Nombre de la función	ft_itoa
Prototipo	char *ft_itoa(int n);
Entregar archivos	-
Parámetros	n: el número entero a convertir.
Valor de retorno	La cadena que representa el número entero. NULL si la asignación falla.
Funciones externas.	malloc
Descripción	Asigna (con malloc(3)) y devuelve una cadena representando el número entero recibido como argumento. Se deben manejar números negativos.

Nombre de la función	ft_strmapi
Prototipo	char *ft_strmapi(char const *s, char (*f)(sin firmar) int, char));
Entregar archivos	-
Parámetros	s: La cadena sobre la cual iterar. f: La función a aplicar a cada carácter.
Valor de retorno	La cadena creada a partir de las sucesivas aplicaciones. apagado'. Devuelve NULL si falla la asignación.
Funciones externas.	malloc
Descripción	Aplica la función 'f' a cada carácter del cadena 's' y pasando su índice como primer argumento para crear una nueva cadena (con malloc(3)) resultante de sucesivas aplicaciones de 'f'.

Nombre de la función	ft_striteri
Prototipo	void ft_striteri(char *s, void (*f)(int sin signo, carbonizarse*));
Entregar archivos	-
Parámetros	s: La cadena sobre la cual iterar. f: La función a aplicar a cada carácter.
Valor de retorno	Ninguno
Funciones externas.	Ninguno
Descripción	Aplica la función 'f' en cada carácter de la cadena pasada como argumento, pasando su índice como primer argumento. Cada personaje pasa por dirección a 'f' que se modificará si es necesario.

Nombre de la función	ft_putchar_fd
Prototipo	vacío ft_putchar_fd(char c, int fd);
Entregar archivos	-
Parámetros	c: El carácter a generar. fd: El descriptor de archivo en el que escribir.
Valor de retorno	Ninguno
Funciones externas.	escribir
Descripción	Envía el carácter 'c' al archivo dado descriptor.

Nombre de la función	ft_putstr_fd
Prototipo	void ft_putstr_fd(char *s, int fd);
Entregar archivos	-
Parámetros	s: la cadena a generar. fd: El descriptor de archivo en el que escribir.
Valor de retorno	Ninguno
Funciones externas.	escribir
Descripción	Envía la cadena 's' al archivo dado descriptor.

Nombre de la función	ft_putendl_fd
Prototipo	void ft_putendl_fd(char *s, int fd);
Entregar archivos	-
Parámetros	s: la cadena a generar. fd: El descriptor de archivo en el que escribir.
Valor de retorno	Ninguno
Funciones externas.	escribir
Descripción	Envía la cadena 's' al descriptor de archivo dado seguido de una nueva línea.

Nombre de la función	ft_putnbr_fd
Prototipo	vacío ft_putnbr_fd(int n, int fd);
Entregar archivos	-
Parámetros	n: el número entero a generar. fd: El descriptor de archivo en el que escribir.
Valor de retorno	Ninguno
Funciones externas.	escribir
Descripción	Envía el número entero 'n' al archivo dado descriptor.

Capítulo IV

parte extra

Si completaste la parte obligatoria, no dudes en ir más allá haciendo esta adicional. Traerá puntos de bonificación si se aprueba con éxito.

Las funciones para manipular memoria y cadenas son muy útiles. Pero pronto descubrirás que manipular listas es aún más útil.

Debe utilizar la siguiente estructura para representar un nodo de su lista. Agregue su declaración a su archivo libft.h:

```
estructura typedef {          lista_s

                              *contenido;
    estructura vacía s_list   *próximo;
}                             lista_t;
```

Los miembros de la estructura t_list son:

• contenido: Los datos contenidos en el nodo.
void * permite almacenar cualquier tipo de datos.

• next: La dirección del siguiente nodo, o NULL si el siguiente nodo es el último.

En su Makefile, agregue una regla de bonificación make para agregar las funciones de bonificación a su libft.a.

La parte de bonificación sólo se valorará si la parte obligatoria es PERFECTA. Perfecto significa que la parte obligatoria se ha realizado íntegramente y funciona sin funcionar mal. Si no has pasado TODOS los requisitos obligatorios, tu parte de bonificación no será evaluada en absoluto.

Implemente las siguientes funciones para poder utilizar fácilmente sus listas.

Nombre de la función	ft_lstnuevo
Prototipo	t_list *ft_lstnew(void *contenido);
Entregar archivos	-
Parámetros	contenido: el contenido con el que crear el nodo.
Valor de retorno	El nuevo nodo
Funciones externas.	malloc
Descripción	Asigna (con malloc(3)) y devuelve un nuevo nodo. La variable miembro 'contenido' se inicializa con el valor del parámetro 'contenido'. La variable 'siguiente' se inicializa en NULL.

Nombre de la función	ft_lstadd_front
Prototipo	void ft_lstadd_front(t_list **lst, t_list *nuevo);
Entregar archivos	-
Parámetros	lst: La dirección de un puntero al primer enlace de una lista. nuevo: la dirección de un puntero al nodo que se va a agregado a la lista.
Valor de retorno	Ninguno
Funciones externas.	Ninguno
Descripción	Agrega el nodo 'nuevo' al principio de la lista.

Nombre de la función	ft_lstsize
Prototipo	int ft_lstsize(t_list *lst);
Entregar archivos	-
Parámetros	lst: El comienzo de la lista.
Valor de retorno	La longitud de la lista.
Funciones externas.	Ninguno
Descripción	Cuenta el número de nodos en una lista.

Nombre de la función	ft_lstlast
Prototipo	t_list *ft_lstlast(t_list *lst);
Entregar archivos	-
Parámetros	lst: El comienzo de la lista.
Valor de retorno	Último nodo de la lista
Funciones externas.	Ninguno
Descripción	Devuelve el último nodo de la lista.

Nombre de la función	ft_lstadd_back
Prototipo	void ft_lstadd_back(t_list **lst, t_list *nuevo);
Entregar archivos	-
Parámetros	lst: La dirección de un puntero al primer enlace de una lista. nuevo: la dirección de un puntero al nodo que se va a agregado a la lista.
Valor de retorno	Ninguno
Funciones externas.	Ninguno
Descripción	Agrega el nodo 'nuevo' al final de la lista.

Nombre de la función	ft_lstdelone
Prototipo	vacío ft_lstdelone(t_list *lst, vacío (*del)(vacío *));
Entregar archivos	-
Parámetros	lst: El nodo a liberar. del: la dirección de la función utilizada para eliminar el contenido.
Valor de retorno	Ninguno
Funciones externas.	gratis
Descripción	Toma como parámetro un nodo y libera la memoria de el contenido del nodo usando la función 'del' dada como parámetro y liberar el nodo. la memoria de 'siguiente' no debe liberarse.

Nombre de la función	ft_lstclear
Prototipo	vacío ft_lstclear(t_list **lst, vacío (*del)(vacío *));
Entregar archivos	-
Parámetros	lst: la dirección de un puntero a un nodo. del: la dirección de la función utilizada para eliminar el contenido del nodo.
Valor de retorno	Ninguno
Funciones externas.	gratis
Descripción	Elimina y libera el nodo dado y cada sucesor de ese nodo, usando la función 'del' y gratis(3). Finalmente, el puntero a la lista debe establecerse en NULO.

Nombre de la función	ft_lstiter
Prototipo	void ft_lstiter(t_list *lst, void (*f)(void *));
Entregar archivos	-
Parámetros	lst: la dirección de un puntero a un nodo. f: La dirección de la función utilizada para iterar la lista.
Valor de retorno	Ninguno
Funciones externas.	Ninguno
Descripción	Itera la lista 'lst' y aplica la función. 'f' sobre el contenido de cada nodo.

Nombre de la función	ft_lstmap
Prototipo	t_list *ft_lstmap(t_list *lst, void *(*f)(void *), vacío (*del)(vacío *));
Entregar archivos	-
Parámetros	lst: la dirección de un puntero a un nodo. f: La dirección de la función utilizada para iterar la lista. del: la dirección de la función utilizada para eliminar el contenido de un nodo si es necesario.
Valor de retorno	La nueva lista. NULL si la asignación falla.
Funciones externas.	malloc, gratis
Descripción	Itera la lista 'lst' y aplica la función. 'f' sobre el contenido de cada nodo. Crea un nuevo lista resultante de las sucesivas aplicaciones de la función 'f'. La función 'del' se utiliza para elimine el contenido de un nodo si es necesario.

Capítulo V

Presentación y evaluación por pares

Entregue su tarea en su repositorio de Git como de costumbre. Durante la defensa sólo se evaluará el trabajo dentro de su repositorio. No dude en volver a verificar los nombres de sus archivos para asegurarse de que sean correctos.

Coloque todos sus archivos en la raíz de su repositorio.