Libft

sua primeira livraria

*Resumo: Este projeto consiste em programar uma biblioteca em C.
Sua biblioteca terá um monte de funções de propósito geral nas quais você construirá
seus programas.*

Versão: 15

Índice Geral

Capítulo I

Introdução

Programar em C pode ser tedioso quando não se tem acesso às funções mais comuns das bibliotecas mais comuns. Este projeto permitirá que você entenda como essas funções funcionam, como implementá-las e o que fazer com elas. Ao criar sua própria biblioteca, os projetos C a seguir serão mais fáceis.

Certifique-se de espalhar sua libft por todo o seu cursus. No entanto, ao usar sua biblioteca, certifique-se de que todas as funções usadas por sua biblioteca respeitem as permitidas por cada projeto.

Capítulo II

Instruções gerais

- Seu projeto deve ser escrito seguindo o Padrão. Se você tiver arquivos ou funções adicionais, eles serão incluídos na verificação de política e você receberá um 0 se houver algum erro de política dentro.
- Suas funções não devem terminar inesperadamente (segfault, erro de barramento, double free, etc) ou ter comportamento indefinido. Se isso acontecer, seu projeto será considerado não funcional e você receberá um 0 durante a avaliação.
- Toda a memória alocada no heap deve ser liberada adequadamente quando necessário. cesariana. Vazamentos de memória não serão permitidos.
- Se o assunto exigir, você deve entregar um Makefile que compilará seus arquivos de origem para a saída necessária com os sinalizadores -Wall, -Werror e -Wextra, claro que seu Makefile não deve revincular.
- Seu Makefile deve conter pelo menos as regras $(NAME), all, clean, fclean e ré.
- Para entregar os bônus do seu projeto, você deve incluir uma regra de bônus em seu Makefile, na qual você adicionará todos os cabeçalhos, bibliotecas ou funções que são proibidas na parte principal do projeto. Os bônus devem estar em arquivos diferentes _bonus.{c/h}. A parte obrigatória e os bônus são avaliados separadamente.
- Se seu projeto permite o uso de libft, você deve copiar sua fonte e seus Makefiles associados em um diretório libft com seu Makefile correspondente. O Makefile do seu projeto deve primeiro compilar a biblioteca usando seu Makefile e, em seguida, compilar o projeto.
- Recomendamos que você crie programas de teste para seu projeto, embora este trabalho **não seja entregue ou avaliado.** Isso lhe dará a oportunidade de verificar se seu programa funciona corretamente durante sua avaliação e de outros colegas.
  E sim, você tem permissão para usar esses testes durante sua avaliação ou de outros colegas.
- Entregue seu trabalho no repositório Git atribuído. Apenas o trabalho do seu repositório Git será avaliado. Se a Deepthought avaliar seu trabalho, fará isso depois de seus colegas. Se um erro for encontrado durante a avaliação do Deepthought, a avaliação será encerrada.

Capítulo III

parte obrigatória

nome do programa	libft.a
arquivos entre de forma alguma	Makefile, libft.h, *.c
Makefile	NOME, tudo, limpo, fclean, re
Funções autorizadas	detalhes abaixo
Se permite usar libft	você ainda não tem
Descrição	Escreva sua própria biblioteca: um conjunto de funções que serão úteis em todo o aula.

III.1. Considerações técnicas

- É proibido *declarar* variáveis globais.
- Se você precisar separar uma função complexa em várias, certifique-se de usar a palavra-chave estática para ela. Dessa forma, as funções permanecerão no arquivo apropriado.
- Coloque todos os seus arquivos na raiz do seu repositório.
- A entrega de arquivos não utilizados é proibida.
- Todos os arquivos .c devem ser compilados com os sinalizadores -Wall -Werror -Wextra.
- Você deve usar o comando ar para gerar a biblioteca. O uso de libtool permanece proibido.
- Seu libft.a deve ser criado na raiz do repositório.

III.2. Parte 1 - funções libc

Para começar, você precisará refazer algumas funções da libc. Suas funções terão os mesmos protótipos e implementarão os mesmos comportamentos das funções originais. Eles devem ser como descrito no homem. A única diferença será a nomenclatura.
Eles começarão com o prefixo “ft_”. Por exemplo, strlen se tornará ft_strlen.

Algumas funções possuem a palavra “restringir” em seus protótipos.
Esta palavra faz parte do padrão c99. Portanto, incluí-lo em seus próprios protótipos é proibido, assim como compilar seu código com o sinalizador -std=c99.

Você precisará escrever suas próprias funções implementando as seguintes funções originais. Não requerem funções autorizadas:

- isalfa
- é dígito
- isalnum
- isascii
- arrancada
- forte
- conjunto de itens
- Zero
- memcpy
- memmove
- strlcpy
- strlcat
- toupeira
- abaixar
- strchr
- strrchr
- strncmp
- memchr
- memcmp
- strnstr
- atoi

Para implementar essas outras duas funções, você precisará usar malloc():

- calloc
- strdup

III.3. Parte 2 - Recursos Adicionais

Nesta segunda parte, você terá que desenvolver um conjunto de funções que, ou não
a biblioteca libc, ou são, mas de uma maneira diferente.

Algumas das seguintes funções podem ser feitas mais
fácil se você usar funções da parte 1.

nome da função	ft_substr
Protótipo	char *ft_substr(char const *s, unsigned int start, tamanho_t len);
arquivos entre de forma alguma	-
Parâmetros	s: A string a partir da qual criar a substring. start: O índice do caractere em 's' do qual iniciar a substring. len: O comprimento máximo da substring.
valor de retorno	A substring resultante. NULL se a reserva de memória falhar.
Funções autorizadas zadas	malloc
Descrição	Reserve (com malloc(3)) e retorne uma substring de la string 's'. A substring começa a partir do índice 'start' e tem um comprimento máximo 'len'.

nome da função	ft_strjoin
Protótipo	char *ft_strjoin(char const *s1, char const *s2);
arquivos entre de forma alguma	-
Parâmetros	s1: A primeira string. s2: A string a ser adicionada a 's1'.
valor de retorno	A nova corda. NULL se a reserva de memória falhar.
Funções autorizadas	malloc
Descrição	Reserve (com malloc(3)) e retorne um novo string, formada pela concatenação de 's1' e 's2'.

nome da função	ft_strtrim
Protótipo	char *ft_strtrim(char const *s1, char const *set);
arquivos entre de forma alguma	-
Parâmetros	s1: A string que deve ser cortada. set: Os caracteres a serem removidos da string.
valor de retorno	A corda cortada. NULL se a reserva de memória falhar.
Funções autorizadas	malloc
Descrição	Remova todos os caracteres da string 'set' do início e do fim de 's1', até encontre um personagem que não pertença a 'set'. o string resultante é retornada com um fallback de malloc(3)

nome da função	ft_split
Protótipo	char **ft_split(char const *s, char c);
arquivos entre de forma alguma	-
Parâmetros	s: A string a ser separada. c: O caractere delimitador.
valor de retorno	A matriz de novas strings resultantes da separação. NULL se a reserva de memória falhar.
Funções autorizadas	malloc, grátis
Descrição	Reserve (usando malloc(3)) um array de strings resultante da separação da string 's' em substrings usando o caractere 'c' como delimitador. o array deve terminar com um ponteiro NULL.

nome da função	ft_itoa
Protótipo	char *ft_itoa(int n);
arquivos entre de forma alguma	-
Parâmetros	n: o inteiro a ser convertido.
valor de retorno	A string que representa o número. NULL se a reserva de memória falhar.
Funções autorizadas	malloc
Descrição	Usando malloc(3), gere uma string que representam o valor inteiro recebido como argumento. Números negativos precisam ser gerenciados.

nome da função	ft_strmapi
Protótipo	char *ft_strmapi(char const *s, char (*f)(unsigned int, caractere));
arquivos entre de forma alguma	-
Parâmetros	s: A string que será repetida. f: La función a aplicar sobre cada carácter.
valor de retorno	A string criada após o uso correto de 'f' em cada personagem. NULL se a reserva de memória falhar.
Funções autorizadas	malloc
Descrição	Para cada caractere da string 's', aplique o função 'f' dando como parâmetros o índice de cada caractere dentro de 's' e o próprio caractere. Gerar uma nova string com o resultado do uso sucessivo o 'f'

nome da função	ft_striteri
Protótipo	void ft_striteri(char *s, void (*f)(unsigned int, Caracteres*));
arquivos entre de forma alguma	-
Parâmetros	s: A string que será repetida. f: La función a aplicar sobre cada carácter.
valor de retorno	Nada
Funções autorizadas	Nenhum
Descrição	Para cada caractere da string 's', aplique a função 'f' dando como parâmetros o índice de cada caractere dentro de 's' e o endereço do próprio caractere, que pode ser modificado se necessário.

nome da função	ft_putchar_fd
Protótipo	void ft_putchar_fd(char c, int fd);
arquivos entre de forma alguma	-
Parâmetros	c: O caractere a ser enviado. fd: O descritor de arquivo para gravar.
valor de retorno	Nada
Funções autorizadas zadas	escrever
Descrição	Envie o caractere 'c' para o descritor de arquivo especificado.

nome da função	ft_putstr_fd
Protótipo	void ft_putstr_fd(char *s, int fd);
arquivos entre de forma alguma	-
Parâmetros	s: A string a ser enviada. fd: O descritor de arquivo para gravar.
valor de retorno	Nada
Funções autorizadas	escrever
Descrição	Enviar string 's' para descritor de arquivo especificado.

nome da função	ft_putendl_fd
Protótipo	void ft_putendl_fd(char *s, int fd);
arquivos entre de forma alguma	-
Parâmetros	s: A string a ser enviada. fd: O descritor de arquivo para gravar.
valor de retorno	Nada
Funções autorizadas	escrever
Descrição	Envie a string 's' para o descritor de arquivo fornecido, seguido por uma quebra de linha.

nome da função	ft_putnbr_fd
Protótipo	void ft_putnbr_fd(int n, int fd);
arquivos entre de forma alguma	-
Parâmetros	n: O número a ser enviado. fd: O descritor de arquivo para gravar.
valor de retorno	Nada
Funções autorizadas	escrever
Descrição	Envie o número 'n' para o descritor de arquivo fornecido.

Capítulo IV

Parte bônus

Se você completar a parte obrigatória, sinta-se à vontade para ir mais longe fazendo esta parte extra. Ele lhe dará pontos extras se você completá-lo corretamente.

As funções para manipular memória e strings são muito úteis... Mas logo você descobrirá Você verá que a manipulação de listas é ainda mais.

Você precisará usar a seguinte estrutura para representar um nó em sua lista. Adicionar a declaração para o seu arquivo libft.h:

```
estrutura typedef      s_list
{
   vazio               *contente;
   struct s_list       *Next; lista_t;
}
```

Os membros da estrutura t_list são:

- conteúdo: as informações contidas no nó.
  void *: permite guardar qualquer tipo de informação.
- next: O endereço do próximo nó, ou NULL se o próximo nó for o último.

Em seu Makefile, adicione uma regra de bônus make que incorpore as funções de bônus em seu libft.a.

A parte bônus será avaliada exclusivamente se a parte obrigatória for perfeita. perfeito? Sim: todos os requisitos da parte obrigatória devem ser preenchidos corretamente. De outro modo,

seus bônus não serão avaliados.

Implemente as seguintes funções para usar facilmente suas listas.

nome da função	ft_lstnew
Protótipo	t_list *ft_lstnew(void *conteúdo);
arquivos entre de forma alguma	-
Parâmetros	content: o conteúdo com o qual criar o nó.
valor de retorno	o novo nó
Funções autorizadas	malloc
Descrição	Crie um novo nó usando malloc(3). o variável de membro 'content' é inicializada com o conteúdo do parâmetro 'conteúdo'. A variável 'próximo', com NULL.

nome da função	ft_lstadd_front
Protótipo	void ft_lstadd_front(t_list **lst, t_list *new);
arquivos entre de forma alguma	-
Parâmetros	lst: o endereço de um ponteiro para o primeiro nó de uma lista. new: um ponteiro para o nó a ser adicionado ao início de a lista.
valor de retorno	Nada
Funções autorizadas	Nenhum
Descrição	Adicione o nó 'novo' ao início da lista 'lst'.

nome da função	ft_lstsize
Protótipo	int ft_lstsize(t_list *lst);
arquivos entre de forma alguma	-
Parâmetros	lst: o início da lista.
valor de retorno	O comprimento da lista.
Funções autorizadas	Nenhum
Descrição	Conta o número de nós em uma lista.

nome da função	ft_lstlast
Protótipo	t_list *ft_lstlast(t_list *lst);
arquivos entre de forma alguma	-
Parâmetros	lst: o início da lista.
valor de retorno	Último nó da lista.
Funções autorizadas	Nenhum
Descrição	Retorna o último nó da lista.

nome da função	ft_lstadd_back
Protótipo	void ft_lstadd_back(t_list **lst, t_list *new);
arquivos entre de forma alguma	-
Parâmetros	lst: O ponteiro para o primeiro nó de uma lista. new: o ponteiro para um nó para adicionar à lista.
valor de retorno	Nada
Funções autorizadas	Nenhum
Descrição	Adicione o nó 'novo' ao final da lista 'lst'.

nome da função	ft_lstdelone
Protótipo	void ft_lstdelone(t_list *lst, void (*del)(void *));
arquivos entre de forma alguma	-
Parâmetros	lst: o nó a ser liberado. del: um ponteiro para a função usada para liberar o conteúdo do nó.
valor de retorno	Nada
Funções autorizadas	gratuitamente
Descrição	Pega um nó 'lst' como parâmetro e libera o memória do conteúdo usando a função 'del' dado como parâmetro, além de liberar o nó. o A memória 'próxima' não deve ser liberada.

nome da função	ft_lstclear
Protótipo	void ft_lstclear(t_list **lst, void (*del)(void *));
arquivos entre de forma alguma	-
Parâmetros	lst: O endereço de um ponteiro para um nó. del: um ponteiro de função usado para excluir o conteúdo de um nó.
valor de retorno	Nada
Funções autorizadas	gratuitamente
Descrição	Remova e libere o nó 'lst' fornecido e todos consecutivas a partir desse nó, usando a função 'del' e livre(3). No final, o ponteiro para a lista deve ser NULL.

nome da função	ft_lstiter
Protótipo	void ft_lstiter(t_list *lst, void (*f)(void *));
arquivos entre de forma alguma	-
Parâmetros	lst: um ponteiro para o primeiro nó. f: um ponteiro para a função que cada nó irá usar.
valor de retorno	Nada
Funções autorizadas	Nenhum
Descrição	Itere a lista 'lst' e aplique a função 'f' no conteúdo de cada nó.

nome da função	ft_lstmap
Protótipo	t_list *ft_lstmap(t_list *lst, void *(*f)(void *), void(*del)(void*));
arquivos entre de forma alguma	-
Parâmetros	lst: um ponteiro para um nó. f: o endereço de um ponteiro para uma função usada na iteração de cada elemento da lista. del: um ponteiro de função usado para excluir o conteúdo de um nó, se necessário.
valor de retorno	A nova lista. NULL se a reserva de memória falhar.
Funções autorizadas	malloc, grátis
Descrição	Itere a lista 'lst' e aplique a função 'f' ao conteúdo de cada nó. Criar uma lista resultante da correta e sucessiva aplicação da função 'f' sobre cada nó. A função 'del' é usada para remover o conteúdo de um nó, se você fizer isso falta.

Capítulo V

Entrega e avaliação

Confirme seu projeto em seu repositório Git como de costume. Somente o trabalho entregue ao repositório será avaliado durante a defesa. Sinta-se à vontade para verificar os nomes dos arquivos várias vezes para verificar se estão corretos.

Deixe todos os seus arquivos na raiz do repositório.