Libft

Sua primeira biblioteca própria

Resumo:
Este projeto é sobre a codificação de uma biblioteca C.
Ela conterá muitas funções de propósito geral nas quais seus programas se apoiarão.

Versão: 15

Conteúdo

Capítulo I

Introdução

A programação C pode ser muito entediante quando não se tem acesso às funções padrão altamente úteis. Este projeto trata de entender a forma como estas funções funcionam, implementando e aprendendo a usá-las. Sua vontade é criar sua própria biblioteca. Será útil já que você a utilizará em suas próximas tarefas escolares em C.

Reserve um tempo para expandir sua `libft` ao longo do ano. Entretanto, ao trabalhar em um novo projeto, não se esqueça de garantir que as funções usadas em sua biblioteca sejam permitidas nas diretrizes do projeto.

Capítulo II

Instruções comuns

- Seu projeto deve ser escrito em C.

- Seu projeto deve ser escrito de acordo com a Norma. Se você tiver arquivos/funções bônus, eles serão incluídos na verificação da norma e você receberá um 0 se houver um erro de norma dentro.

- Suas funções não devem ser abandonadas inesperadamente (falha de segmentação, erro no ônibus, duplo livre, etc.), além de comportamentos indefinidos. Se isto acontecer, seu projeto será considerado não funcional e receberá um 0 durante a avaliação.

- Todo o espaço de memória alocado deve ser devidamente liberado quando necessário. Nenhum vazamento será tolerado.

- Se o assunto o exigir, você deve apresentar um Makefile que compilará seus arquivos fonte para a saída exigida com as bandeiras -Wall, -Wextra e -Werror, use cc, e seu Makefile não deve religar.

- Seu Makefile deve ao menos conter as regras $(NOME), tudo, limpo, limpo, limpo e
  re.

- Para entregar bônus ao seu projeto, você deve incluir um bônus de regra ao seu Makefile, que adicionará todos os vários cabeçalhos, librairies ou funções que são proibidas na parte principal do projeto. Os bônus devem estar em um arquivo diferente _bonus.{c/h} se o assunto não especificar mais nada. A avaliação obrigatória e da parte bônus é feita separadamente.

- Se seu projeto permite que você utilize sua libft, você deve copiar suas fontes e seu Makefile associado em uma pasta libft com seu Makefile associado. O Makefile de seu projeto deve compilar a biblioteca usando seu Makefile e, em seguida, compilar o projeto.

- Nós o encorajamos a criar programas de teste para seu projeto, mesmo que este trabalho **não tenha que ser submetido e não seja classificado**. Ele lhe dará a oportunidade de testar facilmente seu trabalho e o trabalho de seus colegas. Você achará esses testes especialmente úteis durante sua defesa. De fato, durante a defesa, você é livre para usar seus testes e/ou os testes dos colegas que você está avaliando.

- Submeta seu trabalho a seu repositório de git designado. Somente o trabalho no repositório de git - tory será classificado. Se o Deepthought for designado para

classificar seu trabalho, ele será feito

após suas avaliações pelos pares. Se ocorrer um erro em qualquer seção de seu trabalho durante a avaliação do Deepthought, a avaliação será interrompida.

Capítulo III

Parte Obrigatória

Nome do programa	libft.a
Entregue os arquivos	Makefile, libft.h, ft_*.c
Makefile	NOME, tudo, limpo, limpo, limpo, re
Funcionalidades externas.	Detalhado abaixo
Libft autorizada	n/d
Descrição	Escreva sua própria biblioteca:umacoleção de funções que será uma ferramenta útil para o seu cursus.

III.1 Considerações técnicas

- A declaração de variáveis globais é proibida.
- Se você precisar de funções de ajuda para dividir uma função mais complexa, defina-as como funções estáticas. Desta forma, seu escopo será limitado ao arquivo apropriado.
- Coloque todos os seus arquivos na raiz do seu repositório.
- A entrega de arquivos não utilizados é proibida.
- Cada arquivo .c deve ser compilado com as bandeiras -Wall -Wextra -Werror.
- Você deve usar o comando ar para criar sua biblioteca. O uso do comando libtool é proibido.
- Sua libft.a tem que ser criada na raiz de seu repositório.

III.2 Parte - Funções 1Libc

Para começar, você deve refazer um conjunto de funções a partir da libc. Suas funções terão os mesmos protótipos e implementarão os mesmos comportamentos que os originais. Eles devem cumprir com a forma como são definidos em seu homem. A única diferença serão seus nomes. Eles começarão com o prefixo 'ft_'. Por exemplo, strlen se torna ft_strlen.

Alguns dos protótipos de funções que você tem que refazer usam o qualificador "restrito". Estapalavra-chave faz parte do padrão c99. Portanto, é proibido incluí-la em seus próprios protótipos e compilar seu código com a bandeira -std=c99.

Você deve escrever sua própria função implementando as seguintes funções originais. Elas não exigem nenhuma função externa:

- isalpha
- isdigit
- isalnum
- isascii
- isprint
- strlen
- memset
- bzero
- memcpy
- memmove
- strlcpy
- strlcat
- toupper
- tolower
- strchr
- strrchr
- strncmp
- memchr
- memcmp
- strnstr
- atoi

A fim de implementar as duas funções a seguir, você usará malloc():

- calloc
- strdup

III.3 Parte -2 Funções adicionais

Nesta segunda parte, você deve desenvolver um conjunto de funções que ou não estão na libc, ou que são parte dela, mas de uma forma diferente.

Algumas das seguintes funções podem ser úteis para escrever as funções da Parte 1.

Nome da função	ft_substr
Protótipo	char *ft_substr(char const *s, unsigned int start, size_t len);
Entregue os arquivos	-
Parâmetros	s: O fio a partir do qual se cria o substrato. start: O índice de início do substrato na corda 's'. len: O comprimento máximo do substrato.
Valor de retorno	O substrato. NULL se a alocação falhar.
Funcionalidades externas.	malloc
Descrição	Aloca (com malloc(3)) e devolve um substrato do fio 's'. O substrato começa no índice 'start' e é de tamanho máximo 'len'.

Nome da função	ft_strjoin
Protótipo	char *ft_strjoin(char const *s1, char const *s2);
Entregue os arquivos	-
Parâmetros	s1:Acadeia de prefixos. s2:Osufixo de corda.
Valor de retorno	O novo fio. NULL se a alocação falhar.
Funcionalidades externas.	malloc
Descrição	Aloca (com malloc(3)) e devolve um novo que é o resultado da concatenação de 's1' e 's2'.

Nome da função	ft_strtrim
Protótipo	char *ft_strtrim(char const *s1, char const *set);
Entregue os arquivos	–
Parâmetros	s1:Ofio a ser aparado. conjunto:Oconjunto de referência de caracteres a serem aparados.
Valor de retorno	O fio aparado. NULL se a alocação falhar.
Funcionalidades externas.	malloc
Descrição	Aloca (com malloc(3)) e devolve uma cópia de s1' com os caracteres especificados em 'set' removidos do início e do fim da corda.

Nome da função	ft_split
Protótipo	char **ft_split(char const *s, char c);
Entregue os arquivos	–
Parâmetros	s: O fio a ser dividido. c: O caráter delimitador.
Valor de retorno	O conjunto de novas cordas resultantes da divisão. NULL se a alocação falhar.
Funcionalidades externas.	malloc, grátis
Descrição	Aloca (com malloc(3)) e devolve uma matriz de cordas obtidas pela divisão de 's usando o caracter 'c' como delimitador. A matriz deve terminar com um ponteiro NULL.

Nome da função	ft_itoa
Protótipo	char *ft_itoa(int n);
Entregue os arquivos	–
Parâmetros	n:ointeiro a converter.
Valor de retorno	O fio que representa o número inteiro. NULL se a alocação falhar.
Funcionalidades externas.	malloc
Descrição	Aloca (com malloc(3)) e devolve um fio representando o inteiro recebido como argumento. Números negativos devem ser tratados.

Nome da função	ft_strmapi
Protótipo	char *ft_strmapi(char const *s, char (*f)(não assinado int, char));
Entregue os arquivos	–
Parâmetros	s: O fio sobre o qual se deve iterar. f: A função a ser aplicada a cada personagem.
Valor de retorno	A cadeia criada a partir das sucessivas aplicações de 'f'. Devolve o NULL se a alocação falhar.
Funcionalidades externas.	malloc
Descrição	Aplica a função 'f' a cada caráter do string 's', e passando seu índice como primeiro argumento para criar uma nova string (com malloc(3)) resultante de sucessivas aplicações de 'f'.

Nome da função	ft_striteri
Protótipo	void ft_striteri(char *s, void (*f)(unigned int, char*));
Entregue os arquivos	–
Parâmetros	s: O fio sobre o qual se deve iterar. f: A função a ser aplicada a cada personagem.
Valor de retorno	Nenhum
Funcionalidades externas.	Nenhum
Descrição	Aplica a função 'f' em cada caractere de a cadeia passou como argumento, passando seu índice como primeiro argumento.Cadacaractere é passado por endereço a 'f' para ser modificado, se necessário.

Nome da função	ft_putchar_fd
Protótipo	void ft_putchar_fd(char c, int fd);
Entregue os arquivos	–
Parâmetros	c: O caráter a produzir. fd:Odescritor de arquivo no qual escrever.
Valor de retorno	Nenhum
Funcionalidades externas.	escreva
Descrição	Produz o caractere 'c' para o arquivo em questão descritor.

Nome da função	ft_putstr_fd
Protótipo	void ft_putstr_fd(char *s, int fd);
Entregue os arquivos	–
Parâmetros	s: O fio para a saída. fd:Odescritor de arquivo no qual escrever.
Valor de retorno	Nenhum
Funcionalidades externas.	escreva
Descrição	Produz a string 's' para o arquivo dado descritor.

Nome da função	ft_putendl_fd
Protótipo	void ft_putendl_fd(char *s, int fd);
Entregue os arquivos	–
Parâmetros	s: O fio para a saída. fd:Odescritor de arquivo no qual escrever.
Valor de retorno	Nenhum
Funcionalidades externas.	escreva
Descrição	Produz a string 's' para o descritor de arquivo dado seguido de uma nova linha.

Nome da função	ft_putnbr_fd
Protótipo	void ft_putnbr_fd(int n, int fd);
Entregue os arquivos	–
Parâmetros	n:Onúmero inteiro a ser produzido. fd:Odescritor de arquivo no qual escrever.
Valor de retorno	Nenhum
Funcionalidades externas.	escreva
Descrição	Produz o número inteiro 'n' para o arquivo em questão descritor.

Capítulo IV Parte

Bônus

Se você completou a parte obrigatória, não hesite em ir mais longe, fazendo esta parte extra. Isso trará pontos de bônus se for aprovado com sucesso.

As funções para manipular a memória e as cordas são muito úteis. Mas logo você descobrirá que manipular listas é ainda mais útil.

Você tem que usar a seguinte estrutura para representar um nó de sua lista. Acrescente sua declaração ao seu arquivo libft.h:

```
typedef structs_list
{
  nulo*conteúdo    ;
  estrutura s_list*next        ;
                  }t_list;
```

- nulo * permite o armazenamento de qualquer tipo de dado.
- a seguir: O endereço do próximo nó, ou NULL se o próximo nó for o último.

Em seu Makefile, adicione uma regra de fazer bônus para adicionar as funções de bônus à sua libft.a.

A parte bônus só será avaliada se a parte obrigatória for PERFEITA. Perfeito significa que a parte obrigatória foi feita integralmente e funciona sem avarias. Se você não tiver aprovado TODOS os requisitos obrigatórios, sua parte bônus não será avaliada de forma alguma.

Implemente as seguintes funções a fim de utilizar facilmente suas listas.

Nome da função	ft_lstnew
Protótipo	t_list *ft_lstnew(void *content);
Entregue os arquivos	–
Parâmetros	conteúdo:Oconteúdo com o qual se cria o nó.
Valor de retorno	O novo nó
Funcionalidades externas.	malloc
Descrição	Aloca (com malloc(3)) e devolve um novo nó. A variável membro 'conteúdo' é inicializada com o valor do parâmetro 'conteúdo'. A variável 'next' é inicializada para NULL.

Nome da função	ft_lstadd_front
Protótipo	void ft_lstadd_front(t_list **lst, t_list *new);
Entregue os arquivos	–
Parâmetros	lst: O endereço de um ponteiro para o primeiro link de uma lista. novo:Oendereço de um ponteiro para o nó a ser adicionado à lista.
Valor de retorno	Nenhum
Funcionalidades externas.	Nenhum
Descrição	Acrescenta o nó 'novo' no início da lista.

Nome da função	ft_lstsize
Protótipo	int ft_lstsize(t_list *lst);
Entregue os arquivos	–
Parâmetros	lst: O início da lista.
Valor de retorno	A extensão da lista
Funcionalidades externas.	Nenhum
Descrição	Conta o número de nós de uma lista.

Nome da função	ft_lstlast
Protótipo	t_list *ft_lstlast(t_list *lst);
Entregue os arquivos	–
Parâmetros	lst: O início da lista.
Valor de retorno	Último nó da lista
Funcionalidades externas.	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);
Entregue os arquivos	–
Parâmetros	lst: O endereço de um ponteiro para o primeiro link de uma lista. novo:Oendereço de um ponteiro para o nó a ser adicionado à lista.
Valor de retorno	Nenhum
Funcionalidades externas.	Nenhum
Descrição	Acrescenta o nó "novo" no final da lista.

Nome da função	ft_lstdelone
Protótipo	void ft_lstdelone(t_list *lst, void (*del)(void *));
Entregue os arquivos	–
Parâmetros	lst: O nó para liberar. del:Oendereço da função utilizada para apagar o conteúdo.
Valor de retorno	Nenhum
Funcionalidades externas.	grátis
Descrição	Toma como parâmetro um nó e liberta a memória de o conteúdo do nó usando a função 'del' dada como parâmetro e liberar o nó. A memória do 'próximo' não deve ser liberada.

Nome da função	ft_lstclear
Protótipo	void ft_lstclear(t_list **lst, void (*del)(void *));
Entregue os arquivos	–
Parâmetros	lst: O endereço de um ponteiro para um nó. del:Oendereço da função utilizada para excluir o conteúdo do nó.
Valor de retorno	Nenhum
Funcionalidades externas.	grátis
Descrição	Deleta e liberta o nó dado e cada sucessor daquele nó, usando a função 'del' e livre(3). Finalmente, o ponteiro para a lista deve ser definido como NULL.

Nome da função	ft_lstiter
Protótipo	void ft_lstiter(t_list *lst, void (*f)(void *));
Entregue os arquivos	–
Parâmetros	lst: O endereço de um ponteiro para um nó. f: O endereço da função usada para iterar na lista.
Valor de retorno	Nenhum
Funcionalidades externas.	Nenhum
Descrição	Itera a lista 'lst' e aplica a função "f" sobre o conteúdo de cada nó.

Nome da função	ft_lstmap
Protótipo	t_list *ft_lstmap(t_list *lst, void *(*f)(void *), void (*del)(void *));
Entregue os arquivos	–
Parâmetros	lst: O endereço de um ponteiro para um nó. f: O endereço da função usada para iterar na lista. del:Oendereço da função utilizada para excluir o conteúdo de um nó, se necessário.
Valor de retorno	A nova lista. NULL se a alocação falhar.
Funcionalidades externas.	malloc, grátis
Descrição	Itera a lista 'lst' e aplica a função "f" sobre o conteúdo de cada nó.Criauma nova lista resultante das sucessivas aplicações da função 'f'. A função 'del' é usada para excluir o conteúdo de um nó, se necessário.

Capítulo V

Submissão e avaliação por pares

Entregue sua tarefa em seu repositório Git, como de costume. Somente o trabalho dentro de seu repositório - itório será avaliado durante a defesa. Não hesite em verificar novamente os nomes de seus arquivos para garantir que estejam corretos.

Coloque todos os seus arquivos na raiz do seu repositório.