Libft

Sua primeira biblioteca própria

Resumo:
Este projeto trata da codificação de uma biblioteca C.
Ele conterá muitas funções de uso geral nas quais seus programas dependerão.

Versão: 16

Conteúdo

Capítulo I

Introdução

A programação C pode ser muito tediosa quando não se tem acesso às funções padrão altamente úteis. Este projeto trata de entender como essas funções funcionam, implementá-las e aprender a utilizá-las. Você criará sua própria biblioteca. Será útil, pois você o usará em suas próximas tarefas escolares C.

Aproveite o tempo para expandir seu libft ao longo do ano. Porém, ao trabalhar em um novo projeto, não se esqueça de garantir que as funções utilizadas em sua biblioteca são permitidas nas diretrizes do projeto.

Capítulo II

Instruções Comuns

- Seu projeto deve ser escrito em C.
- Seu projeto deverá ser escrito de acordo com a Norma. Se você tiver arquivos/funções bônus, eles serão incluídos na verificação de norma e você receberá 0 se houver um erro de norma.
- Suas funções não devem encerrar inesperadamente (falha de segmentação, erro de barramento, liberação dupla, etc.) além de comportamentos indefinidos. Caso isso aconteça, seu projeto será considerado não funcional e receberá nota 0 na avaliação.
- Todo o espaço de memória alocado no heap deve ser liberado adequadamente quando necessário. Sem vazamentos será tolerado.
- Se o assunto exigir, você deve enviar um Makefile que irá compilar seus arquivos fonte para a saída necessária com as flags -Wall, -Wextra e -Werror, use cc, e seu Makefile não deve relinkar.
- Seu Makefile deve conter pelo menos as regras $(NAME), all, clean, fclean e ré.
- Para entregar bônus ao seu projeto, você deve incluir uma regra bônus em seu Makefile, que adicionará todos os diversos cabeçalhos, bibliotecas 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 da parte obrigatória e da parte bônus é feita separadamente.
- Se o seu projeto permite que você use sua libft, você deve copiar seus fontes e seu Makefile associado em uma pasta libft com seu Makefile associado. O Makefile do seu projeto deve compilar a biblioteca usando seu Makefile e depois compilar o projeto.
- Nós encorajamos você a criar programas de teste para o seu projeto, mesmo que este trabalho **não precise ser enviado e não receba notas.** Isso lhe dará a chance de testar facilmente seu trabalho e o de seus colegas. Você achará esses testes especialmente úteis durante sua defesa. Na verdade, durante a defesa, você é livre para usar seus testes e/ou os testes do colega que está avaliando.
- Envie seu trabalho para o repositório git designado. Somente o trabalho no repositório git será avaliado. Se Deepthought for designado para avaliar seu trabalho, isso será feito

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

Capítulo III

Parte obrigatória

Nome do programa	libft.a
Entregar arquivos	Makefile, libft.h, ft_*.c
Funções	NOME, tudo, limpo, fclean, re
externas do Makefile.	Detalhado abaixo
Libft autorizado	n / D
Descrição	Escreva sua própria biblioteca: uma coleção de funções essa será uma ferramenta útil para o seu curso.

III.1 Considerações técnicas

- É proibido declarar variáveis globais.
- Se você precisar de funções auxiliares para dividir uma função mais complexa, defina-as como estáticas funções. Desta forma, seu escopo ficará limitado ao arquivo apropriado.
- Coloque todos os seus arquivos na raiz do seu repositório.
- É proibido entregar arquivos não utilizados.
- Todos os arquivos .c devem ser compilados com as flags -Wall -Wextra -Werror.
- Você deve usar o comando ar para criar sua biblioteca. Usando o comando libtool é proibido.
- Seu libft.a deve ser criado na raiz do seu repositório.

III.2 Parte 1 - Funções Libc

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

Alguns dos protótipos de funções que você precisa refazer usam o qualificador 'restrict'. Esta palavra-chave faz parte do padrão c99. Portanto, é proibido incluí-lo em seus próprios protótipos e compilar seu código com a flag -std=c99.

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

- isalfa
- também
- a sala de gelo
- isascii
- corrida
- estrelinha
- conjunto de memórias
- zero
- memcpy
- memove
- strlcpy
- strlcat
- parte superior
- abaixar
- strchr
- strrhr
- strncmp
- memchr
- memcmp
- strstr
- reboque

Para implementar as duas funções a seguir, você usará malloc():

- chamada
- forte

III.3 Parte 2 - Funções adicionais

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

Algumas das funções a seguir podem ser úteis para escrever o funções da Parte 1.

Nome da função	ft_substr
Protótipo	char *ft_substr(char const *s, início int não assinado, tamanho_t comprimento);
Entregar arquivos	-
Parâmetros	s: A string a partir da qual criar a substring. start: O índice inicial da substring no cordas'. len: O comprimento máximo da substring.
Valor de retorno	A substring. NULL se a alocação falhar.
Funções externas.	maloc
Descrição	Aloca (com malloc(3)) e retorna uma substring da string 's'. A substring 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);
Entregar arquivos	-
Parâmetros	s1: a sequência de prefixos. s2: a string do sufixo.
Valor de retorno	A nova sequência. NULL se a alocação falhar.
Funções externas.	maloc
Descrição	Aloca (com malloc(3)) e retorna um novo string, 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);
Entregar arquivos	-
Parâmetros	s1: A string a ser cortada. set: O conjunto de referência de caracteres a serem cortados.
Valor de retorno	A corda aparada. NULL se a alocação falhar.
Funções externas.	maloc
Descrição	Aloca (com malloc(3)) e retorna uma cópia de 's1' com os caracteres especificados em 'set' removidos desde o início e o fim da string.

Nome da função	ft_split
Protótipo	char **ft_split(char const *s, char c);
Entregar arquivos	-
Parâmetros	s: A string a ser dividida. c: O caractere delimitador.
Valor de retorno	A matriz de novas cadeias de caracteres resultantes da divisão. NULL se a alocação falhar.
Funções externas.	Malloc, grátis
Descrição	Aloca (com malloc(3)) e retorna um array de strings obtidas pela divisão de 's' usando o caractere 'c' como delimitador. A matriz deve terminar com um ponteiro NULL.

Nome da função	ft_itoa
Protótipo	char *ft_itoa(int n);
Entregar arquivos	-
Parâmetros	n: o número inteiro a ser convertido.
Valor de retorno	A string que representa o número inteiro. NULL se a alocação falhar.
Funções externas.	maloc
Descrição	Aloca (com malloc(3)) e retorna uma string 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));
Entregar arquivos	-
Parâmetros	s: a string na qual iterar. f: A função a ser aplicada a cada personagem.
Valor de retorno	A string criada a partir dos aplicativos sucessivos desligado'. Retorna NULL se a alocação falhar.
Funções externas.	maloc
Descrição	Aplica a função 'f' a cada caractere do string 's' e passando seu índice como primeiro argumento para criar uma nova string (com malloc(3)) resultante de aplicações sucessivas de 'f'.

Nome da função	ft_stritteri
Protótipo	void ft_striteri(char *s, void (*f)(unsigned int, Caracteres*));
Entregar arquivos	-
Parâmetros	s: a string na qual iterar. f: A função a ser aplicada a cada personagem.
Valor de retorno	Nenhum
Funções externas.	Nenhum
Descrição	Aplica a função 'f' em cada caractere de a string passada como argumento, passando seu índice como primeiro argumento. Cada personagem é passado endereço para 'f' para ser modificado se necessário.

Nome da função	ft_putchar_fd
Protótipo	void ft_putchar_fd(char c, int fd);
Entregar arquivos	-
Parâmetros	c: O caractere a ser gerado. fd: O descritor de arquivo no qual escrever.
Valor de retorno	Nenhum
Funções externas.	escrever
Descrição	Produz o caractere 'c' para o arquivo fornecido descritor.

Nome da função	ft_putstr_fd
Protótipo	void ft_putstr_fd(char *s, int fd);
Entregar arquivos	-
Parâmetros	s: a string a ser gerada. fd: O descritor de arquivo no qual escrever.
Valor de retorno	Nenhum
Funções externas.	escrever
Descrição	Produz a string 's' para o arquivo fornecido descritor.

Nome da função	ft_putendl_fd
Protótipo	void ft_putendl_fd(char *s, int fd);
Entregar arquivos	-
Parâmetros	s: a string a ser gerada. fd: O descritor de arquivo no qual escrever.
Valor de retorno	Nenhum
Funções externas.	escrever
Descrição	Produz a string 's' para o descritor de arquivo fornecido seguido por uma nova linha.

Nome da função	ft_putnbr_fd
Protótipo	void ft_putnbr_fd(int n, int fd);
Entregar arquivos	-
Parâmetros	n: O número inteiro a ser gerado. fd: O descritor de arquivo no qual escrever.
Valor de retorno	Nenhum
Funções externas.	escrever
Descrição	Produz o número inteiro 'n' para o arquivo fornecido descritor.

Capítulo IV

Parte bônus

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

Funções para manipular memória e strings são muito úteis. Mas você logo descobrirá que manipular listas é ainda mais útil.

Você deve usar a seguinte estrutura para representar um nó da sua lista. Adicione sua declaração ao seu arquivo libft.h:

```
estrutura typedef {          lista_s

                             *contente;
  estrutura vazia s_list     *próximo;
}                            lista_t;
```

Os membros da estrutura t_list são:

- conteúdo: Os dados contidos no nó. void * permite armazenar qualquer tipo de dados.
- next: O endereço do próximo nó, ou NULL se o próximo nó for o último.

No seu Makefile, adicione uma regra make bônus para adicionar as funções de bônus ao seu 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 mau funcionamento. Se você não passou em TODOS os requisitos obrigatórios, sua parte do bônus não será avaliada de forma alguma.

Implemente as seguintes funções para usar facilmente suas listas.

Nome da função	ft_lstnew
Protótipo	t_list *ft_lstnew(void *conteúdo);
Entregar arquivos	-
Parâmetros	content: o conteúdo com o qual criar o nó.
Valor de retorno	O novo nó
Funções externas.	maloc
Descrição	Aloca (com malloc(3)) e retorna um novo nó. A variável de membro 'content' é inicializada com o valor do parâmetro 'conteúdo'. A variável 'next' é inicializado como NULL.

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

Nome da função	ft_lstsize
Protótipo	int ft_lstsize(lista_t *lst);
Entregar arquivos	-
Parâmetros	lst: O início da lista.
Valor de retorno	O comprimento da lista
Funções externas.	Nenhum
Descrição	Conta o número de nós em uma lista.

Nome da função	ft_lstúltimo
Protótipo	lista_t *ft_lstlast(lista_t *lst);
Entregar arquivos	-
Parâmetros	lst: O início da lista.
Valor de retorno	Último nó da lista
Funções 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);
Entregar arquivos	-
Parâmetros	lst: O endereço de um ponteiro para o primeiro link de uma lista. new: O endereço de um ponteiro para o nó a ser adicionado à lista.
Valor de retorno	Nenhum
Funções externas.	Nenhum
Descrição	Adiciona o nó 'novo' no final da lista.

Nome da função	ft_lstdelone
Protótipo	void ft_lstdelone(t_list *lst, void (*del)(void *));
Entregar arquivos	-
Parâmetros	lst: O nó a ser liberado. del: O endereço da função usada para excluir o conteúdo.
Valor de retorno	Nenhum
Funções externas.	livre
Descrição	Toma como parâmetro um nó e libera a memória do o conteúdo do nó usando a função 'del' fornecida como parâmetro e libere o nó. A memória de 'próximo' não deve ser liberado.

Nome da função	ft_lstclear
Protótipo	void ft_lstclear(t_list **lst, void (*del)(void *));
Entregar arquivos	-
Parâmetros	lst: O endereço de um ponteiro para um nó. del: O endereço da função usada para excluir o conteúdo do nó.
Valor de retorno	Nenhum
Funções externas.	livre
Descrição	Exclui e libera o nó fornecido e todos sucessor desse nó, usando a função 'del' e gratuito(3). Finalmente, o ponteiro para a lista deve ser definido como NULO.

Nome da função	ft_lstiter
Protótipo	void ft_lstiter(t_list *lst, void (*f)(void *));
Entregar arquivos	-
Parâmetros	lst: O endereço de um ponteiro para um nó. f: O endereço da função usada para iterar a lista.
Valor de retorno	Nenhum
Funções externas.	Nenhum
Descrição	Itera a lista 'lst' e aplica a função 'f' no conteúdo de cada nó.

Nome da função	ft_lstmap
Protótipo	lista_t *ft_lstmap(lista_t *lst, void *(*f)(void *), vazio (*del)(vazio *));
Entregar arquivos	-
Parâmetros	lst: O endereço de um ponteiro para um nó. f: O endereço da função usada para iterar a lista. del: O endereço da função usada para excluir o conteúdo de um nó, se necessário.
Valor de retorno	A nova lista. NULL se a alocação falhar.
Funções externas.	Malloc, grátis
Descrição	Itera a lista 'lst' e aplica a função 'f' no conteúdo de cada nó. Cria um novo lista resultante das sucessivas aplicações de a função 'f'. A função 'del' é usada para exclua o conteúdo de um nó, se necessário.

Capítulo V

Envio e avaliação por pares

Entregue sua tarefa em seu repositório Git normalmente. Somente os trabalhos dentro do seu repositório serão avaliados durante a defesa. Não hesite em verificar os nomes dos seus arquivos para garantir que estão corretos.

Coloque todos os seus arquivos na raiz do seu repositório.

Rnpu cebwrpg bs gur 97 Pbzzba Pber pbagnvaf na rapbqrq uvag. Sbe rnpu pvepyr, bayl bar cebwrpg cebivqrf gur pbeerpg uvag arrqrq sbe gur arkg pvepyr. Guvf punyyratr vf vaqvivqhny, gurer vf bayl n cevmr sbe bar fghqrag jvaare cebivqvat nyy qrpbqrq zrffntrf. Nal nqinagntrq crbcyr pna cynl, yvxr pheerag be sbezre fgnss, ohg gur cevmr jvyy erznva flzobyvp. Gur uvag sbe guvf svefg cebwrpg vf: Ynetr pbjf trarebfvgl pbzrf jvgu punegf naq sbhe oybaqr ungf gb qrsl hccre tenivgl ureb