Libft

Sua primeira biblioteca

Sumário: Este projeto consiste em codificar uma biblioteca em `C`*.*
Ela conterá muitas funções de uso geral das quais seus programas dependerão.

Versão: 16.1

Conteúdo

Capítulo I

Introdução

Programar em `C` pode ser muito tedioso quando não se tem acesso às funções padrão altamente úteis. Este projeto é sobre entender como essas funções funcionam, implementá-las e aprender a usá-las. Você criará sua própria biblioteca. Será útil, pois você a usará em suas próximas tarefas em `C`.

Tome o tempo para expandir sua `libft` ao longo do ano. No entanto, 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

Regras gerais

- O seu projeto deve ser escrito em C.
- O seu projeto deve estar codificado dentro da Norma. Se você possui arquivos ou funções bônus, elas serão incluídas na verificação da norma e você receberá `0` no projeto se não seguir a norma.
- Suas funções não devem parar inesperadamente (falha de segmentação, erro bus, double free, etc.), exceto no caso de um comportamento indefinido. Se isso acontecer, o seu projeto será considerado não funcional e você receberá um `0` na avaliação.
- Qualquer memória alocada no heap deve ser liberada quando necessário. Nenhum leak será tolerado.
- Se o projeto pedir, você deve fazer um `Makefile` que compilará as suas fontes para criar a saída solicitada, utilizando as sinalizações `-Wall`, `-Wextra` e `-Werror`. O seu Makefile não deve ter relink.
- Se o seu projeto pedir um Makefile, o seu Makefile deve no mínimo conter as regras (NAME), `all`, `clean`, `fclean` e `re`.
- Para entregar o bônus, você deve incluir uma regra `bonus` no seu Makefile que vai adicionar os diversos headers, bibliotecas e funções que não são autorizadas na parte principal do projeto. Os bônus devem ficar em um arquivo `_bonus.{c/h}`. A avaliação da parte obrigatória e e da parte bônus são feitas separadamente.
- Se o projeto autorizar o uso do seu `libft`, você deve copiar suas fontes e o seu `Makefile` associado em uma pasta `libft` na raiz. O `Makefile` do seu projeto deve compilar a biblioteca usando o `Makefile` dela, depois compilar o projeto.
- Nós recomendamos criar programas de teste para o seu projeto, mesmo que esse trabalho **não seja entregue nem avaliado**. Isso te dará uma chance de testar facilmente o seu trabalho assim como o dos seus colegas. Isso será especialmente útil durante a sua avaliação. Inclusive, durante a avaliação, você fica livre para usar seus testes e/ou os testes da pessoa que você está avaliando.

- Você deve entregar o seu trabalho no repositório git que lhe foi atribuído. Somente o trabalho colocado no repositório git será avaliado. Se o Deepthought precisar corrigir o seu trabalho, isso será feito no fim do processo das avaliações dos colegas. Se um erro acontecer durante a avaliação Deepthought, ela será finalizada.

Capítulo III

Parte Mandatória

Nome do programa	`libft.a`
Arquivos para entregar	`Makefile, libft.h, ft_*.c`
Makefile	`NAME, all, clean, fclean, re`
Funções externas autorizadas	`Detalhadas abaixo`
Libft autorizado	`n/a`
Descrição	`Escreva sua própria biblioteca: uma coleção de funções que serão ferramentas úteis para o seu curso.`

III.1 Considerações Técnicas

- Declarar variáveis globais é proibido.
- Se você precisa usar uma subfunção em mais de uma função, você deve declará-la como `static` para limitar seu escopo ao arquivo em que ela é declarada.
- 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 utilizar o comando `ar` para criar sua biblioteca. O uso do comando `libtool` é proibido.
- Seu `libft.a` deve ser criado na raiz do seu repositório. Your `libft.a` has to be created at the root of your repository.

III.2 Parte 1 - Funções da LibC

Para começar, você deve reescrever um conjunto de funções da `libc`. Suas funções terão os mesmos protótipos e implementarão os mesmos comportamentos que as originais. Elas devem estar em conformidade com a forma como são definidas em seus `man`. A única diferença será seus nomes. Elas começarão com o prefixo '`ft_`'. Por exemplo, `strlen` torna-se `ft_strlen`.

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

Você deve escrever suas próprias funções implementando as funções originais a seguir. Elas não precisam de nenhuma função externas:

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

Para implementar cada uma das próximas duas funções, você usará `malloc()`:

- `calloc`
- `strdup`

Dependendo do seu sistema operacional, a página do man da calloc e o comportamento da função podem ser diferentes. A instrução a seguir sobrescreve o que você pode encontrar na página man: Se nmemb ou size é 0, então calloc() retorna um único valor de ponteiro que depois pode ser passado com sucesso para free().

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 destas 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);
Ficheiros para entregar	-
Parâmetros	s: A string que servirá como fonte para a criação da substring. start: O índice inicial onde começa a substring da string 's'. len: O comprimento máximo da substring.
Valor de retorno	A substring. NULL se a alocação falhar.
Funções externas autorizadas	malloc
Descrição	Aloca memória (com malloc(3)) e retorna uma substring da string 's'. A substring começa no índice 'start' e tem o tamanho máximo de 'len'.

Nome da função	ft_strjoin
Protótipo	char *ft_strjoin(char const *s1, char const *s2);
Ficheiros para entregar	-
Parâmetros	s1: A string prefixa. s2: A string sufixa.
Valor de retorno	A nova string. NULL se a alocação falhar.
Funções externas autorizadas	malloc
Descrição	Aloca (with malloc(3)) e retorna uma nova 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);
Ficheiros para entregar	-
Parâmetros	s1: A string a ser aparada. set: O conjunto de caracteres para aparar.
Valor de retorno	A string aparada. NULL se a alocação falhar.
Funções externas autorizadas	malloc
Descrição	Aloca (with malloc(3)) e retorna uma cópia de 's1', com os caracteres especificados em 'set' removidos do início e do final da string.

Nome da função	ft_split
Protótipo	char **ft_split(char const *s, char c);
Ficheiros para entregar	-
Parâmetros	s: A string a ser dividida. c: O caractere delimitador.
Valor de retorno	O array de strings resultants da divisão. NULL se a alocação falhar.
Funções externas autorizadas	malloc, free
Descrição	Aloca (com malloc(3)) e retorna um array de strings obtidas pela divisão de 's', que utilizou o caractere 'c' como delimitador. O array deve terminar com um ponteiro NULL.

Nome da função	ft_itoa
Protótipo	char *ft_itoa(int n);
Ficheiros para entregar	-
Parâmetros	n: O inteiro a ser convertido.
Valor de retorno	A string representando o integer. NULL se a alocação falhar.
Funções externas autorizadas	malloc
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)(unsigned int, char));
Ficheiros para entregar	-
Parâmetros	s: A string a ser iterada. f: A função a ser aplicada a cada caractere.
Valor de retorno	A string criada a partir de sucessivas aplicações de 'f'. NULL se a alocação falhar.
Funções externas autorizadas	malloc
Descrição	Aplica a função 'f' para cada caractere da string 's', e passa 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)(unsigned int, char*));
Ficheiros para entregar	-
Parâmetros	s: A string a ser iterada. f: A função a ser aplicada em cada caractere.
Valor de retorno	Nenhum
Funções externas autorizadas	Nenhuma
Descrição	Aplica a função 'f' em cada caractere de uma string passada como argumento, passando seu índice como primeiro argumento. Cada caractere é passado por endereço para 'f' para serem modificados se necessário.

Nome da função	ft_putchar_fd
Protótipo	void ft_putchar_fd(char c, int fd);
Ficheiros para entregar	-
Parâmetros	c: O caractere a ser escrito. fd: O file descriptor no qual o caracter será escrito.
Valor de retorno	Nenhum
Funções externas autorizadas	write
Descrição	Tem como saída o caractere 'c' para o file descriptor especificado.

Nome da função	`ft_putstr_fd`
Protótipo	`void ft_putstr_fd(char *s, int fd);`
Ficheiros para entregar	-
Parâmetros	`s: A string a ser escrita.` `fd: O file descriptor no qual a string será escrita.`
Valor de retorno	Nenhum
Funções externas autorizadas	`write`
Descrição	Tem como saída a string 's' para o file descriptor especificado.

Nome da função	`ft_putendl_fd`
Protótipo	`void ft_putendl_fd(char *s, int fd);`
Ficheiros para entregar	-
Parâmetros	`s: A string a ser escrita.` `fd: O file descriptor no qual a string será escrita.`
Valor de retorno	Nenhum
Funções externas autorizadas	`write`
Descrição	Tem como saída a string 's' para o file descriptor especificado, seguida por uma quebra de linha..

Nome da função	`ft_putnbr_fd`
Protótipo	`void ft_putnbr_fd(int n, int fd);`
Ficheiros para entregar	-
Parâmetros	`n: O inteiro a ser escrito.` `fd: O file descriptor no qual o inteiro será escrito.`
Valor de retorno	Nenhum
Funções externas autorizadas	`write`
Descrição	Tem como saída o inteiro 'n' para o file descriptor especificado.

Capítulo IV

Parte bônus

Se você completou a parte obrigatória, não hesite em avançar fazendo esta parte adicional. Ela trará pontos extras se sua implementação for bem sucedida.

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

Você deverá utilizar a seguinte estrutura para representar os elementos (nós) de sua lista. Adicione sua declaração ao seu arquivo `libft.h`:

```
typedef struct      s_list
{
  void              *content;
  struct s_list     *next;
}                   t_list;
```

Os membros da struct `t_list` são:

- `content`: A informação contida no nó.
  `void *` permite que o nó armazene qualquer tipo de dados.
- `next`: O endereço do próximo nó da lista, ou `NULL` se o próximo nó for o último.

No seu Makefile, adicione uma regra `make bonus` para adicionar as funções bônus à sua `libft.a`.

A parte bônus apenas será avaliada se a parte obrigatória estiver PERFEITA. Perfeita significa que a parte obrigatória foi integralmente feita e funciona sem falhas. Se você não passou em TODOS os requisitos obrigatórios, sua parte bônus não será avaliada.

Implemente as seguintes funções para manipular suas listas com facilidade.

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

Nome da função	`ft_lstadd_front`
Protótipo	`void ft_lstadd_front(t_list **lst, t_list *new);`
Ficheiros para entregar	-
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 autorizadas	Nenhuma
Descrição	Adiciona o nó 'new' no início da lista.

Nome da função	`ft_lstsize`
Protótipo	`int ft_lstsize(t_list *lst);`
Ficheiros para entregar	-
Parâmetros	`lst:` O início da lista.
Valor de retorno	O tamanho da lista.
Funções externas autorizadas	Nenhuma
Descrição	Conta o número de nós de uma lista.

Nome da função	`ft_lstlast`
Protótipo	`t_list *ft_lstlast(t_list *lst);`
Ficheiros para entregar	-
Parâmetros	`lst: O início da lista.`
Valor de retorno	`O último nó da lista.`
Funções externas autorizadas	`Nenhuma`
Descrição	`Retorna o último nó de uma lista.`

Nome da função	`ft_lstadd_back`
Protótipo	`void ft_lstadd_back(t_list **lst, t_list *new);`
Ficheiros para entregar	-
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 autorizadas	`Nenhuma`
Descrição	`Adiciona o nó 'new' ao final da lista.`

Nome da função	`ft_lstdelone`
Protótipo	`void ft_lstdelone(t_list *lst, void (*del)(void *));`
Ficheiros para entregar	-
Parâmetros	`lst: O nó para ser liberado.` `del: O endereço da função utilizada para deletar o conteúdo..`
Valor de retorno	`Nenhum`
Funções externas autorizadas	`free`
Descrição	`Pega como um parâmetro um nó e libera a memória de seu conteúdo usando a função 'del' dada como um parâmetro e libera o nó. A memória do 'next' não deve ser liberada.`

Nome da função	`ft_lstclear`
Protótipo	`void ft_lstclear(t_list **lst, void (*del)(void *));`
Ficheiros para entregar	-
Parâmetros	`lst: O endereço de um ponteiro para um nó.` `del: O endereço da função usada para deletar o conteúdo do nó.`
Valor de retorno	`Nenhum`
Funções externas autorizadas	`free`
Descrição	`Deleta e libera o nó especificado e todos os sucessores deste nó, utilizando a função 'del' e free(3).` `No final, o ponteiro para a lista deve ser definido para NULL.`

Nome da função	`ft_lstiter`
Protótipo	`void ft_lstiter(t_list *lst, void (*f)(void *));`
Ficheiros para entregar	-
Parâmetros	`lst: The address of a pointer to a node.` `f: The address of the function used to iterate on the list.`
Valor de retorno	`Nenhum`
Funções externas autorizadas	`Nenhuma`
Descrição	`Itera a lista 'lst' e aplica 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 *));`
Ficheiros para entregar	-

Parâmetros	`lst: O endereço do ponteiro para um nó.`
	`f: O endereço da função usada para iterar na lista.`
	`del: O endereço da função usada para deletar o conteúdo de um nó, caso necessário.`

Valor de retorno	`The new list.` `NULL se a alocação falhar.`
Funções externas autorizadas	`malloc, free`
Descrição	`Itera a lista 'lst' e aplica a função 'f' no conteúdo de cada nó. Cria uma nova lista resultante de sucessivas aplicações da função 'f'. A função 'del' é utilizada para deletar o conteúdo de um nó, caso necessário.`

Capítulo V

Entrega e avaliação entre pares

Entregue seu projeto no repositório `Git` designado.

Apenas o trabalho no seu repositório será avaliado 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