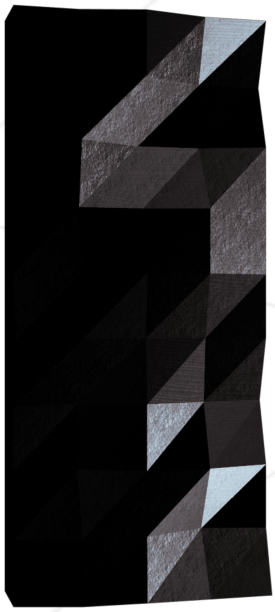

# Libft

Sua primeira biblioteca própria

*Resumo:*

*Este projeto trata da codificação de umCbiblioteca.*

*Ele conterá muitas funções de propósito geral nas quais seus programas se basearão.*

*Versão: 15*

# Conteúdo

# Capítulo I

# Introdução

Ca programação pode ser muito tediosa quando não se tem acesso às funções padrão altamente úteis. Este projeto trata de entender como essas funções funcionam, implementando e aprendendo a usá-las. Você criará sua própria biblioteca. Será útil, pois você o usará em seu próximoCatribuições escolares.

Aproveite o tempo para expandir sualibftao 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

# Instruções comuns

- Seu projeto deve ser escrito em C.
- Seu projeto deve ser escrito de acordo com a Norma. Se você tiver arquivos/funções de bônus, eles serão incluídos na verificação de norma e você receberá um0se houver um erro de norma dentro.
- Suas funções não devem encerrar inesperadamente (falha de segmentação, erro de barramento, double free, etc) além de comportamentos indefinidos. Se isso acontecer, seu projeto será considerado não funcional e receberá um0durante a avaliação.
- Todo o espaço de memória alocado no heap deve ser liberado adequadamente quando necessário. Nenhum vazamento será tolerado.
- Se o assunto exigir, você deve enviar umMakefileque compilará seus arquivos de origem para a saída necessária com os sinalizadores -Parede, -Wextrae -Erro,use cc, e seu Makefile não deve revincular.
- SuaMakefiledeve conter pelo menos as regras $(NAME), tudo, limpo, fcleane ré.
- Para entregar bônus ao seu projeto, você deve incluir uma regrabônusao seu Makefile, que adicionará todos os vários cabeçalhos, bibliotecas ou funções que são proibidas na parte principal do projeto. Os bônus devem estar em um arquivo diferente _bônus.{c/h}se o assunto não especificar mais nada. A avaliação obrigatória e da parte bônus é feita separadamente.
- Se o seu projeto permite que você use seulibft,você deve copiar suas fontes e seus associadosMakefileem umlibftpasta com seu Makefile associado. O seu projeto Makefiledeve compilar a biblioteca usando seuMakefile,então compile o projeto.
- Nós encorajamos você a criar programas de teste para seu projeto mesmo que este trabalho **não terá que ser enviado e não será avaliado**. Isso lhe dará a chance 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 do colega que está avaliando.
- Envie seu trabalho para o repositório git atribuído. Apenas o trabalho no repositório git será avaliado. Se o Deepthought for atribuído 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 |
| **Entregue os arquivos** | Makefile, libft.h, ft_*.c |
| **Makefile** | NOME, tudo, limpo, fclean, re |
| **Funções externas.** | Detalhado abaixo |
| **Libft autorizado** | n / D |
| **Descrição** | Escreva sua própria biblioteca: uma coleção de funções que serão uma ferramenta útil para seu cursus. |

## III.1 Considerações técnicas

- Declarar variáveis globais é proibido.
- Se você precisar de funções auxiliares para dividir uma função mais complexa, defina-as como estático funções. Dessa forma, seu escopo será limitado ao arquivo apropriado.
- Coloque todos os seus arquivos na raiz do seu repositório.
- Entregar arquivos não utilizados é proibido.
- Cada .c fios arquivos devem compilar com os sinalizadores -Parede -Wextra -Werror.
- Você deve usar o comandoarpara criar sua biblioteca. Usando olibtoolcomando é proibido.
- Sualibft.adeve ser criado na raiz do seu repositório.

## III.2 Parte 1 - Funções Libc

Para começar, você deve refazer um conjunto de funções dolibc.Suas funções terão os mesmos protótipos e implementarão os mesmos comportamentos dos originais. Eles devem cumprir a forma como são definidos em seuscara.A única diferença serão seus nomes. Eles começarão com o 'pés_'prefixo. Por exemplo,fortetorna-seft_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 o sinalizador -std=c99.

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

- isalfa
- é dígito
- isalnum
- isascii
- imprint
- forte
- conjunto de memórias
- bzero
- memcpy
- memmove
- strlcpy
- strlcat
- chapéu de coco
- abaixar
- strchr
- strchr
- strncmp
- memchr
- memcmp
- strnstr
- atoi

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

- calloc
- strdup

## III.3 Parte 2 - Funções adicionais

Nesta segunda parte, você deve desenvolver um conjunto de funções que não estão nolibc, ou que fazem 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: A string a partir da qual criar a substring. start: O índice inicial da substring na string 's'.<br><br>len: O comprimento máximo da substring. |
| **Valor de retorno** | A subcadeia.<br>NULL se a alocação falhar. |
| **Funções externas.** | malloc |
| **Descrição** | Aloca (com malloc(3)) e retorna uma substring da string 's'.<br>A substring começa no índice 'start' e tem tamanho máximo 'len'. |

| **Nome da função** | ft_strjoin |
|---|---|
| **Protótipo** | char *ft_strjoin(char const *s1, char const *s2); |
| **Entregue os arquivos** | - |
| **Parâmetros** | s1: A seqüência de prefixo.<br>s2: A cadeia de sufixo. |
| **Valor de retorno** | A nova corda.<br>NULL se a alocação falhar. |
| **Funções externas.** | malloc |
| **Descrição** | Aloca (com 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); |
| **Entregue os arquivos** | - |
| **Parâmetros** | s1: A string a ser cortada.<br>definir: O conjunto de referência de caracteres a serem aparados. |
| **Valor de retorno** | A corda cortada.<br>NULL se a alocação falhar. |
| **Funções externas.** | malloc |
| **Descrição** | Aloca (com malloc(3)) e retorna uma cópia de 's1' com os caracteres especificados em 'set' removidos do início e do fim da string. |

| **Nome da função** | ft_split |
|---|---|
| **Protótipo** | char **ft_split(char const *s, char c); |
| **Entregue os arquivos** | - |
| **Parâmetros** | s: A string a ser dividida. O<br>c: caractere delimitador. |
| **Valor de retorno** | A matriz de novas strings resultantes da divisão. NULL se a alocação falhar. |
| **Funções externas.** | malloc, grátis |
| **Descrição** | Aloca (com malloc(3)) e retorna um array de strings obtido 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); |
| **Entregue os arquivos** | - |
| **Parâmetros** | n: o inteiro a ser convertido. |
| **Valor de retorno** | A string que representa o inteiro. NULL<br>se a alocação falhar. |
| **Funções externas.** | 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)); |
| **Entregue os arquivos** | - |
| **Parâmetros** | s: A string na qual iterar.<br>f: A função a ser aplicada a cada caractere. |
| **Valor de retorno** | A string criada a partir das sucessivas aplicações de 'f'.<br><br>Retorna NULL se a alocação falhar. |
| **Funções externas.** | malloc |
| **Descrição** | Aplica a função 'f' a cada caractere da string 's', 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)(unsigned int, char*)); |
| **Entregue os arquivos** | - |
| **Parâmetros** | s: A string na qual iterar.<br>f: A função a ser aplicada a cada caractere. |
| **Valor de retorno** | Nenhum |
| **Funções externas.** | Nenhum |
| **Descrição** | Aplica a função 'f' em cada caractere da string passada como argumento, passando seu índice como primeiro argumento. Cada caractere é passado por endereço para '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 caractere a ser gerado.<br>fd: O descritor de arquivo no qual gravar. |
| **Valor de retorno** | Nenhum |
| **Funções externas.** | escrever |
| **Descrição** | Gera o caractere 'c' para o descritor de arquivo fornecido. |

| | |
|---|---|
| **Nome da função** | ft_putstr_fd |
| **Protótipo** | void ft_putstr_fd(char *s, int fd); |
| **Entregue os arquivos** | - |
| **Parâmetros** | s: A cadeia de caracteres para saída.<br>fd: O descritor de arquivo no qual gravar. |
| **Valor de retorno** | Nenhum |
| **Funções externas.** | escrever |
| **Descrição** | Gera a string 's' para o descritor de arquivo fornecido. |

| | |
|---|---|
| **Nome da função** | ft_putendl_fd |
| **Protótipo** | void ft_putendl_fd(char *s, int fd); |
| **Entregue os arquivos** | - |
| **Parâmetros** | s: A cadeia de caracteres para saída.<br>fd: O descritor de arquivo no qual gravar. |
| **Valor de retorno** | Nenhum |
| **Funções externas.** | escrever |
| **Descrição** | Gera 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); |
| **Entregue os arquivos** | - |
| **Parâmetros** | n: O inteiro a ser gerado.<br>fd: O descritor de arquivo no qual gravar. |
| **Valor de retorno** | Nenhum |
| **Funções externas.** | escrever |
| **Descrição** | Gera o inteiro 'n' para o descritor de arquivo fornecido. |

# 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 tem que usar a seguinte estrutura para representar um nó da sua lista. Adicione sua declaração ao seulibft.h file:

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

Os membros dot_listestrutura são:

- contente:Os dados contidos no nó. vazio *
  permite armazenar qualquer tipo de dados.
- Next:O endereço do próximo nó, ouNULOse o próximo nó for o último.

Em seu Makefile, adicione umfazer bônusregra para adicionar as funções de bônus ao seulibft.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 falhas. Se você não passou TODAS as
requisitos obrigatórios, sua parte bônus não será avaliada.

Implemente as seguintes funções para usar facilmente suas listas.

| **Nome da função** | ft_lstnew |
|---|---|
| **Protótipo** | t_list *ft_lstnew(void *conteúdo); |
| **Entregue os arquivos** | - |
| **Parâmetros** | content: o conteúdo com o qual criar o nó. |
| **Valor de retorno** | O novo nó |
| **Funções externas.** | 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 como 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.<br>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(t_list *lst); |
| **Entregue os 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_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 |
| **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); |
| **Entregue os arquivos** | - |
| **Parâmetros** | lst: O endereço de um ponteiro para o primeiro link de uma lista.<br>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 *)); |
| **Entregue os arquivos** | - |
| **Parâmetros** | lst: O nó para liberar.<br>del: O endereço da função usada para excluir o conteúdo. |
| **Valor de retorno** | Nenhum |
| **Funções externas.** | gratuitamente |
| **Descrição** | Toma como parâmetro um nó e libera a memória do conteúdo do nó usando a função 'del' dada como parâmetro e libera o nó. A memória de '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. O<br>del: endereço da função usada para excluir o conteúdo do nó. |
| **Valor de retorno** | Nenhum |
| **Funções externas.** | gratuitamente |
| **Descrição** | Exclui e libera o nó fornecido e todos os sucessores desse nó, usando a função 'del' e free(3).<br><br>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.<br>f: O endereço da função usada para iterar na 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** | 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.<br>f: O endereço da função usada para iterar na lista.<br><br>del: O endereço da função usada para excluir o conteúdo de um nó, se necessário. |
| **Valor de retorno** | A nova lista.<br>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 uma 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 seuGitrepositório como de costume. Apenas o trabalho dentro do seu repositório 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.