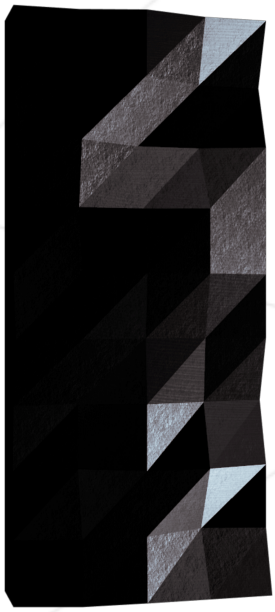

ft_printf

Porque ft_putnbr() e ft_putstr() não são suficientes

Sumário:
O objetivo deste projeto é simples. Você deverá programar o printf().
Você aprenderá, principalmente, a utilizar um número variável de argumentos. Isso não é legal? Isso na verdade é muito legal :)

Versão: 10

Conteúdo

Capítulo I

Introduction

Você descobrirá uma função `C` popular e versátil: `printf()`. Este exercício é uma ótima oportunidade para melhorar suas habilidades de programação. É um exercício de dificuldade moderada.

Você descobrirá **funções variádicas** em `C`.

A chave para um `ft_printf` bem-sucedido é um código bem estruturado e extensível.

`Uma vez que este projeto seja aprovado, você poderá adicionar seu ft_printf() à sua libft para que você possa usá-lo em seus projetos futuros em C.`

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	`libftprintf.a`
Arquivos para entregar	`Makefile, *.h, */*.h, *.c, */*.c`
Makefile	`NAME, all, clean, fclean, re`
Funções externas autorizadas	`malloc, free, write,` `va_start, va_arg, va_copy, va_end`
Libft autorizado	`Sim`
Descrição	`Escreva uma biblioteca que contenha a função` `ft_printf(), que imitará o printf() original`

Você terá que recodificar a função `printf()` da `libc`.

O protótipo do `ft_printf()` é:

```
int     ft_printf(const char *, ...);
```

Aqui estão os requisitos:

- Não implemente o gerenciamento de buffer do `printf()` original.
- Sua função deve lidar com as seguintes conversões: `cspdiuxX%`
- Sua função será comparada com o `printf()` original.
- Você deve usar o comando `ar` para criar sua biblioteca.
  O uso do comando `libtool` é proibido.
- Seu `libftprintf.a` deve ser criado na raiz do seu repositório.

Você terá que implementar as seguintes conversões:

- `%c` Imprime um único caractere.
- `%s` Imprime uma string (conforme definido pela convenção comum do `C`).
- `%p` O endereço do ponteiro `void *` deve ser impresso em formato hexadecimal.
- `%d` Imprime um número decimal (base 10).
- `%i` Imprime um inteiro em base 10.
- `%u` Imprime um número decimal (base 10) sem sinal.
- `%x` Imprime um número em formato hexadecimal (base 16) em minúsculas.
- `%X` Imprime um número em formato hexadecimal (base 16) em maiúsculas.
- `%%` Imprime um sinal de porcentagem.

Capítulo IV

Parte Bônus

Você não precisa fazer todos os bônus.

Lista de bônus:

- Gerencie qualquer combinação dos seguintes flags: '`-0.`' e a largura mínima do campo em todas as conversões.
- Gerencie todos os seguintes flags: '`# +`' (Sim, um deles é um espaço)

Se você deseja fazer o bônus, pense na implementação de suas características extras desde o início. Desta forma, você evitará as armadilhas de uma abordagem ingênua.

A parte bônus só será avaliada se a parte obrigatória estiver PERFEITA. Perfeito significa que todos os requisitos obrigatórios foram cumpridos e funcionam sem falhas. Se você não passou em TODOS os requisitos obrigatórios, sua parte bônus não será avaliada de forma alguma.

Capítulo V

Submissão e Avaliação entre Pares

Entregue seu projeto em seu repositório `Git` como de costume. Apenas o trabalho dentro do seu repositório será avaliado durante a defesa. Não hesite em verificar os nomes de seus arquivos para garantir que estejam corretos.

Uma vez aprovado, você poderá adicionar seu `ft_printf()` à sua `libft` para que você possa usá-lo em seus projetos futuros em C.

```
++++++++++[>+>+++>+++++++>++++++++++<<<<-]>>>.>---.++++++++++++.++.+++
+++.--.<<++.>>------.------------.++++++++++++.<<.>>++++++.------------
.-------.  ++++++++++++++++++.<<.>>----------------.+++++.+++++++++.---
----------.--.+ ++++++++++++++++.--------.++++++++++++.<<.>>----------
-------------.+++.+++ ++++.---.----.++++++++++++++++.---------------
--.-.<<.>>+++++.+++++.<<.>-------...
```