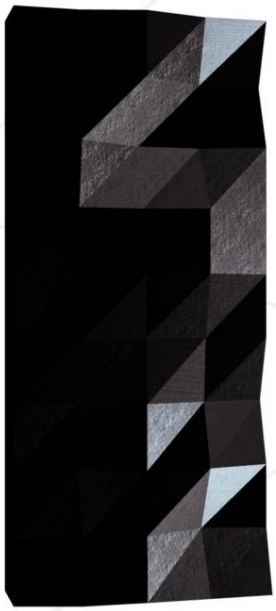

ft_printf

Porque ft_putnbr() e ft_putstr() não são suficientes

Resumo: O
objetivo deste projeto é bastante simples. Você irá recodificar printf().
Você aprenderá principalmente sobre como usar um número variável de argumentos.

Versão: 9

Conteúdo

Capítulo I

Introdução

Você descobrirá uma função C popular e versátil: printf(). Este exercício é uma ótima oportunidade para melhorar suas habilidades de programação. É 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 esta tarefa seja aprovada, você poderá adicionar seu ft_printf() à sua libft para que você possa usá-lo em seus projetos C da escola.

Capítulo II

Instruções comuns

- 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 um 0 se 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 0 durante a 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 compilará seus arquivos de origem para a saída necessária com os sinalizadores -Wall, -Wextra e -Werror, use cc, e seu Makefile não deve revincular.

- Seu Makefile deve conter pelo menos as regras $(NAME), all, clean, fclean e ré.

- Para entregar bônus ao seu projeto, você deve incluir um bônus de regra no 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 _bonus.{c/h}. A avaliação obrigatória e da parte bônus é feita separadamente.

- Se seu projeto permite que você use sua libft, você deve copiar suas 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, em seguida, compilar o projeto.

- Incentivamos você a criar programas de teste para seu projeto, mesmo que este trabalho **não precise ser submetido e não seja 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 designado para avaliar seu trabalho, isso será feito após as avaliações dos colegas. 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	libftprintf.a
Entregue os arquivos	Makefile, *.h, */*.h, *.c, */*.c
Makefile	NOME, tudo, limpo, fclean, re
Funções externas.	malloc, livre, escrever, va_start, va_arg, va_copy, va_end
Libft autorizado	sim
Descrição	Escreva uma biblioteca que contenha ft_printf(), um função que irá imitar o printf() original

Você precisa recodificar a função printf() da libc.

O protótipo de 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 a 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 precisa implementar as seguintes conversões:

- %c Imprime um único caractere.
- %s Imprime uma string (conforme definido pela convenção C comum).
- %p O argumento do ponteiro void * deve ser impresso em formato hexadecimal.
- %d Imprime um número decimal (base 10).
- %i Imprime um inteiro na base 10.
- %u Imprime um número decimal sem sinal (base 10).
- %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 tem que fazer todos os bônus.

Lista de bônus:

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

Se você planeja completar a parte bônus, pense na implementação de seus recursos extras desde o início. Dessa forma, você evitará as armadilhas de uma abordagem ingênua.

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 em TODOS os requisitos obrigatórios, sua parte bônus não será avaliada.

Capítulo V

Submissão e avaliação por pares

Entregue sua tarefa em seu repositório Git 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.

Uma vez que esta tarefa seja aprovada, você poderá adicionar seu ft_printf() à sua libft para que você possa usá-lo em seus projetos C da escola.