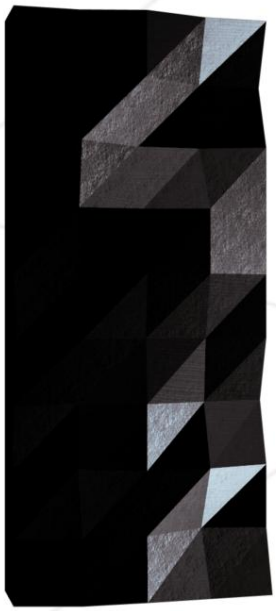

ft_printf

Porque ft_putnbr() e ft_putstr() não são suficientes

Resumo: O
objetivo deste projeto é bastante direto. Você recodificará printf().
Você aprenderá principalmente sobre o uso de um número variável de argumentos. Quão legal é isso??
É realmente muito legal :)

Versão: 9.2

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.

Assim que esta tarefa for aprovada, você poderá adicionar seu ft_printf () ao seu libft para que possa usá-lo em seus projetos C escolares.

Capítulo II

Instruções Comuns

- Seu projeto deve ser escrito em C.
- Seu projeto deve ser redigido de acordo com a Norma. Se você tiver arquivos/funções de bônus, eles serão incluídos na verificação da norma e você receberá um 0 se houver um erro de norma dentro.
- Suas funções não devem ser encerradas 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 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 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 ativar bônus em seu projeto, você deve incluir uma regra de bônus em 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} se o assunto não especificar mais nada. A avaliação da parte obrigatória e 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 o seu projeto, mesmo que este trabalho **não precise ser enviado e não seja avaliado.** Isso lhe dará a chance de testar facilmente seu trabalho e o 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 par que está avaliando.
- Envie seu trabalho para o repositório git atribuído. Somente o trabalho no repositório git será avaliado. Se o Deepthought for designado para avaliar seu trabalho, isso será feito

após suas avaliações de 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	libftprintf.a Makefile,
Entregar arquivos	*.h, */*.h, *.c, */*.c NAME, all, clean, fclean, re
Makefile	malloc, free, write, va_start, va_arg,
Funções externas.	va_copy, va_end Sim
Liberação autorizada	
Descrição	Escreva uma biblioteca que contenha ft_printf(), uma função que imitará a printf() original

Você tem que 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 o printf() original.

• Você deve usar o comando ar para criar sua biblioteca.
Usar o comando libtool é proibido.

• Seu libftprintf.a deve ser criado na raiz do seu repositório.

Você deve 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 no 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 maiúsculo hexadecimal (base 16).
- %% Imprime um sinal de porcentagem.

Capítulo IV

Parte bônus

Você não precisa fazer todos os bônus.

Lista de bônus:

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

Se você planeja concluir 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 peça obrigatória foi executada integralmente e funciona sem avarias. Se você não passou em TODOS os requisitos obrigatórios, sua parte de bônus não será avaliada.

Capítulo V

Submissão e avaliação por pares

Entregue sua atribuição 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.

Assim que esta tarefa for aprovada, você poderá adicionar seu ft_printf () ao seu libft para que possa usá-lo em seus projetos C escolares.