



Obter próxima linha

Ler uma linha de um fd é muito tedioso

Resumo:

Este projeto trata da programação de uma função que retorna uma linha lida de um descritor de arquivo.

Versão: 12

Conteúdo

EU	Metas	2
	II Instruções Comuns	3
	III Parte obrigatória	5
	IV parte bônus	7
	V Submissão e avaliação por pares	8

Capítulo I

Metas

Este projeto não só permitirá que você adicione uma função muito conveniente à sua coleção, mas também fará com que você aprenda um novo conceito muito interessante em programação C: variáveis estáticas.

Capítulo II

Instruções Comuns

- Seu projeto deve ser escrito em C.
- Seu projeto deverá ser escrito de acordo com a Norma. Se você tiver arquivos/funções bônus, eles serão incluídos na verificação de norma e você receberá 0 se houver um erro de norma.
- Suas funções não devem encerrar inesperadamente (falha de segmentação, erro de barramento, liberação dupla, etc.) além de comportamentos indefinidos. Caso isso aconteça, 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 fonte para a saída necessária com as flags -Wall, -Wextra e -Werror, use cc, e seu Makefile não deve relinkar.
- Seu Makefile deve conter pelo menos as regras \$(NAME), all, clean, fclean e ré.
- Para entregar bônus ao seu projeto, você deve incluir uma regra bônus em seu Makefile, que adicionará todos os diversos 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 da parte bônus é feita separadamente.
- Se o seu projeto permitir que você use sua libft, você deve copiar seus 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 depois compilar o projeto.
- Nós encorajamos você a criar programas de teste para o seu projeto, mesmo que este trabalho **não precise ser enviado e não receba notas**. Isso lhe dará a chance de testar facilmente seu trabalho e o de seus colegas. Você achará esses testes especialmente úteis durante sua defesa. Na verdade, 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 designado. Somente o trabalho no repositório git será avaliado. Se Deepthought for designado para avaliar seu trabalho, isso será feito

Obter próxima linha

Ler uma linha de um fd é muito tedioso

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 da função	get_next_line
Protótipo	char *get_next_line(int fd);
Entregar arquivos	get_next_line.c, get_next_line_utils.c, get_next_line.h
Parâmetros	fd: O descritor de arquivo para leitura
Valor de retorno	Linha de leitura: comportamento correto NULL: não há mais nada para ler ou há um erro ocorreu
Funções externas.	ler, malloc, grátis
Descrição	Escreva uma função que retorne uma linha lida de um descritor de arquivo

- Chamadas repetidas (por exemplo, usando um loop) para sua função get_next_line() devem permitir que você lê o arquivo de texto apontado pelo descritor de arquivo, **uma linha por vez**.
- Sua função deverá retornar a linha que foi lida.
Se não houver mais nada para ler ou se ocorreu um erro, deverá retornar NULL.
- Certifique-se de que sua função funcione conforme esperado tanto ao ler um arquivo quanto ao leitura da entrada padrão.
- **Observe** que a linha retornada deve incluir o caractere de terminação \n, exceto se o final do arquivo foi atingido e não termina com um caractere \n.
- Seu arquivo de cabeçalho get_next_line.h deve conter pelo menos o protótipo da função get_next_line().
- Adicione todas as funções auxiliares necessárias no arquivo get_next_line_utils.c.



Um bom começo seria saber o que é uma [variável estática](#) é.

Obter próxima linha

Ler uma linha de um fd é muito tedioso

- Como você terá que ler arquivos em `get_next_line()`, adicione esta opção ao seu chamada do compilador: `-D BUFFER_SIZE=n` Definirá o tamanho do buffer para `read()`.
O valor do tamanho do buffer será modificado pelos seus pares avaliadores e pelo Moulinette para testar seu código.



Devemos ser capazes de compilar este projeto com e sem o sinalizador `-D BUFFER_SIZE` além dos sinalizadores usuais. Você pode escolher o valor padrão de sua preferência.

- Você compilará seu código da seguinte maneira (um tamanho de buffer de 42 é usado como exemplo):
`cc -Wall -Wextra -Werror -D BUFFER_SIZE=42 <arquivos>.c`
- Consideramos que `get_next_line()` tem um comportamento indefinido se o arquivo apontado pelo descritor de arquivo mudou desde a última chamada enquanto `read()` não atingiu o final do arquivo.
- Consideramos também que `get_next_line()` possui um comportamento indefinido ao ler um arquivo binário. No entanto, você pode implementar uma maneira lógica de lidar com esse comportamento, se desejar.



Sua função ainda funciona se o valor `BUFFER_SIZE` for 9999? Se for 1? 10000000?
Você sabe por quê?



Tente ler o mínimo possível cada vez que `get_next_line()` for chamado. Se você encontrar uma nova linha, deverá retornar a linha atual.

Não leia o arquivo inteiro e processe cada linha.

Proibido

- Você não tem permissão para usar sua libft neste projeto.
- `lseek()` é proibido.
- Variáveis globais são proibidas.

Capítulo IV

Parte bônus

Este projeto é simples e não permite bônus complexos. No entanto, confiamos na sua criatividade. Se você completou a parte obrigatória, experimente esta parte bônus.

Aqui estão os requisitos da parte bônus:

- Desenvolva `get_next_line()` usando apenas uma variável estática.
- Seu `get_next_line()` pode gerenciar vários descritores de arquivo ao mesmo tempo.
Por exemplo, se você puder ler os descritores de arquivo 3, 4 e 5, poderá ler de um fd diferente por chamada sem perder o thread de leitura de cada descritor de arquivo ou retornar uma linha de outro fd.
Isso significa que você deve ser capaz de chamar `get_next_line()` para ler fd 3, depois fd 4, depois 5, depois mais uma vez 3, mais uma vez 4 e assim por diante.

Anexe o sufixo `_bonus.[c|h]` aos arquivos da parte bônus.

Isso significa que, além dos arquivos de peças obrigatórios, você entregará os 3 arquivos a seguir:

- `get_next_line_bonus.c`
- `get_next_line_bonus.h`
- `get_next_line_utils_bonus.c`



A parte bônus só será avaliada se a parte obrigatória for **PERFEITA**. Perfeito significa que a parte obrigatória foi feita integralmente e funciona sem mau funcionamento. Se você não passou em **TODOS** os requisitos obrigatórios, sua parte do bônus não será avaliada de forma alguma.

Capítulo V

Envio e avaliação por pares

Entregue sua tarefa em seu repositório Git normalmente. Somente os trabalhos dentro do seu repositório serão avaliados durante a defesa. Não hesite em verificar os nomes dos seus arquivos para garantir que estão corretos.



Ao escrever seus testes, lembre-se que: 1) Tanto o tamanho do buffer quanto o tamanho da linha podem ter valores muito diferentes.

2) Um descritor de arquivo não aponta apenas para arquivos normais. Seja inteligente e verifique com seus colegas. Prepare um conjunto completo de diversos testes para defesa.

Depois de aprovado, não hesite em adicionar `get_next_line()` ao seu libft.



/=ÿ^\\[(_)\$ ^@|V †|-|@_T /-!570@1<|-\\£1_`/ ¢@/^ÿ vv!7}{ ???