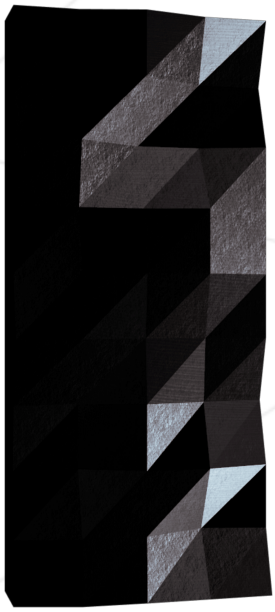

Get Next Line

Ler uma linha de um dfile descriptor é muito tedioso.

Preâmbulo:
Este projeto consiste em programar uma função que retorna uma linha lida de um file descriptor.

Versão: 13

Sumário

Capítulo I

Objetivos

Este projeto não apenas permitirá que você adicione uma função altamente útil à sua coleção, mas também lhe ensinará um conceito importante na programação em `C`: variáveis estáticas.

Capítulo II

Regras gerais

- 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 `0` durante a avaliação.

- Qualquer memória alocada na 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` et `-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` et `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}` se o subject do projeto não especificar algo diferente. A avaliação da parte obrigatória e da parte bônus são feitas separadamente.

- Se o projeto autorizar o uso da sua `libft`, você deve copiar suas fontes e o seu Makefile associado em uma pasta `libft`. O Makefile do seu projeto deve compilar a biblioteca usando o seu Makefile, e 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.

- Você deve entregar o seu trabalho no git que lhe foi atribuído. Somente o trabalho colocado no git será avaliado. Se a Moulinette precisar corrigir o seu trabalho,

isso será feito no fim do processo das avaliações dos colegas. Se um erro acontecer durante a avaliação da Moulinette, ela será finalizada.

Capítulo III

Instruções de IA

Contexto

Este projeto foi desenvolvido para ajudá-lo a descobrir os blocos de construção fundamentais do seu treinamento em TIC.

Para ancorar adequadamente os conhecimentos e habilidades-chave, é essencial adotar uma abordagem criteriosa ao uso de ferramentas e suporte de IA.

A verdadeira aprendizagem fundamental exige um esforço intelectual genuíno — através de desafios, repetição e trocas de aprendizagem entre pares.

Para uma visão geral mais completa de nossa posição sobre a IA — como ferramenta de aprendizagem, como parte do currículo de TIC e como expectativa no mercado de trabalho — consulte as perguntas frequentes dedicadas na intranet.

Mensagem principal

- Construa bases sólidas sem atalhos.
- Desenvolva verdadeiramente habilidades técnicas e de poder.
- Experimente a verdadeira aprendizagem entre pares, comece a aprender como aprender e resolver novos problemas.
- A jornada de aprendizagem é mais importante que o resultado.
- Aprenda sobre os riscos associados à IA e desenvolva práticas eficazes de controle e contramedidas para evitar armadilhas comuns.

Regras para o aluno:

- Você deve aplicar o raciocínio às suas tarefas atribuídas, especialmente antes de recorrer à IA.
- Você não deve pedir respostas diretas à IA.
- Você deve aprender sobre a abordagem global da 42 em relação à IA.

Resultados da fase:

Nesta fase fundamental, você obterá os seguintes resultados:

- Obter bases sólidas em tecnologia e codificação.
- Saber por que e como a IA pode ser perigosa durante esta fase.

Comentários e exemplo:

- Sim, sabemos que a IA existe — e sim, ela pode resolver seus projetos. Mas você está aqui para aprender, não para provar que a IA aprendeu. Não perca seu tempo (nem o nosso) apenas para demonstrar que a IA pode resolver o problema dado.
- Aprender na 42 não é sobre saber a resposta — é sobre desenvolver a capacidade de encontrar uma. A IA lhe dá a resposta diretamente, mas isso o impede de construir seu próprio raciocínio. E o raciocínio leva tempo, esforço e envolve falhas. O caminho para o sucesso não deve ser fácil.
- Lembre-se de que durante os exames, a IA não estará disponível — sem internet, sem smartphones, etc. Você perceberá rapidamente se confiou demais na IA em seu processo de aprendizagem.
- A aprendizagem entre pares o expõe a diferentes ideias e abordagens, melhorando suas habilidades interpessoais e sua capacidade de pensar de forma divergente. Isso é muito mais valioso do que apenas conversar com um bot. Então não seja tímido — converse, faça perguntas e aprenda juntos!
- Sim, a IA fará parte do currículo — tanto como ferramenta de aprendizagem quanto como um tópico em si. Você até terá a chance de construir seu próprio software de IA. Para saber mais sobre nossa abordagem crescente, consulte a documentação disponível na intranet.

✓ Boa prática:

> Estou travado em um novo conceito. Pergunto a alguém próximo como ele abordou isso. Conversamos por 10 minutos — e de repente, clica. Entendi.

✗ Má prática:

> Uso secretamente a IA, copio algum código que parece certo. Durante a avaliação entre pares, não consigo explicar nada. Eu falho. Durante o exame — sem IA — estou travado novamente. Eu falho.

Capítulo IV

Parte Obrigatória

Nome da função	`get_next_line`
Protótipo	`char *get_next_line(int fd);`
Arquivos para entregar	`get_next_line.c, get_next_line_utils.c, get_next_line.h`
Parâmetros	`fd: O file descriptor a ser lido`
Valor de retorno	`Linha lida: comportamento correto` `NULL: não há mais nada para ler ou ocorreu um erro`
Funções externas autorizadas	`read, malloc, free`
Descrição	`Escreva uma função que retorna uma linha lida de um file descriptor`

- Chamadas repetidas (por exemplo, usando um loop) para sua função `get_next_line()` devem permitir que você leia o arquivo de texto apontado pelo file descriptor, **uma linha por vez**.

- Sua função deve retornar a linha que foi lida.
  Se não houver mais nada para ler ou se ocorrer um erro, ela deve retornar `NULL`.

- Certifique-se de que sua função funcione como esperado tanto ao ler um arquivo quanto ao ler da entrada padrão.

- **Observe** que a linha retornada deve incluir o caractere de terminação `\n`, exceto quando o final do arquivo é alcançado e o arquivo não termina com um caractere `\n`.

- Seu header file `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.

- Como você precisará ler arquivos em `get_next_line()`, adicione esta opção à sua chamada do compilador: `-D BUFFER_SIZE=n`
  Isso definirá o tamanho do buffer para `read()`.
  O valor do tamanho do buffer será ajustado por seus avaliadores e pela Moulinette para testar seu código.

Devemos ser capazes de compilar este projeto com e sem a flag -D BUFFER_SIZE além das flags usuais. Você pode escolher qualquer valor padrão que preferir.

- 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`

- `get_next_line()` exibe comportamento indefinido se o arquivo associado ao file descriptor for modificado após a última chamada, enquanto `read()` ainda não atingiu o final do arquivo.

- `get_next_line()` também exibe 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? E se for 1? 10000000? Você sabe por quê?

Leia o mínimo possível de dados a cada vez que get_next_line() for chamada. Se um caractere de nova linha for encontrado, retorne a linha atual imediatamente.
Não leia todo o arquivo e depois processe cada linha.

Proibido

- Você não pode usar sua `libft` neste projeto.

- `lseek()` é proibido.

- Variáveis globais são proibidas.

Capítulo V

Parte Bônus

Este projeto é simples e não suporta recursos de bônus complexos. No entanto, confiamos em sua criatividade. Se você completou a parte obrigatória, considere tentar esta seção de bônus.

Aqui estão os requisitos da parte bônus:

- Desenvolva `get_next_line()` usando apenas uma variável estática.
- Sua `get_next_line()` pode gerenciar vários file descriptors ao mesmo tempo. Por exemplo, se você estiver lendo dos file descriptors 3, 4 e 5, você deve ser capaz de ler de um file descriptor diferente a cada chamada, sem perder o controle do estado de leitura de cada file descriptor ou retornar uma linha de um arquivo diferente. Isso significa que você deve ser capaz de chamar `get_next_line()` para ler do fd 3, depois do fd 4, depois do fd 5, depois novamente do fd 3, depois do fd 4 e assim por diante, sem perder o controle do estado de leitura para cada file descriptor.

Adicione o sufixo `_bonus.[c\h]` aos arquivos da parte bônus.
Isso significa que, além dos arquivos da parte obrigatória, 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 estiver perfeita. "Perfeita"significa que a parte obrigatória foi integralmente feita e funciona sem falhas. Se você não passou TODOS os requisitos obrigatórios, sua parte bônus não será avaliada.

Capítulo VI

Submissão e Avaliação por Pares

Envie sua tarefa em seu repositório `Git` como de costume. Apenas o conteúdo dentro do seu repositório será avaliado durante a defesa. Certifique-se de verificar duas vezes os nomes dos arquivos para garantir a precisão.

```
Ao escrever seus testes, lembre-se de que:
1) Tanto o tamanho do buffer quanto o tamanho da linha podem ter
valores muito diferentes.
2) Um file descriptor não aponta apenas para arquivos regulares.
Seja minucioso e verifique seu trabalho com seus colegas.  Prepare um
conjunto abrangente de testes diversos para a defesa.
```

Após aprovação, não hesite em adicionar seu `get_next_line()` à sua `libft`.

Durante a avaliação, uma breve **modificação do projeto** pode ser ocasionalmente solicitada. Isso pode envolver uma pequena mudança de comportamento, algumas linhas de código para escrever ou reescrever, ou um recurso fácil de adicionar.

Embora esta etapa possa **não ser aplicável a todos os projetos**, você deve estar preparado para ela se for mencionada nas diretrizes de avaliação.

Esta etapa visa verificar sua compreensão real de uma parte específica do projeto. A modificação pode ser realizada em qualquer ambiente de desenvolvimento que você escolher (por exemplo, sua configuração usual), e deve ser viável em poucos minutos — a menos que um prazo específico seja definido como parte da avaliação.
Você pode, por exemplo, ser solicitado a fazer uma pequena atualização em uma função ou script, modificar uma exibição ou ajustar uma estrutura de dados para armazenar novas informações, etc.

Os detalhes (escopo, alvo, etc.) serão especificados nas **diretrizes de avaliação** e podem

variar de uma avaliação para outra para o mesmo projeto.

/=∂/\/\[](_)§ /\/\@|V †|-|@¯|¯ /-/!570®1<|-\£1_`/ ¢@/\/\ε vv!7}{ ???