Guia Explicativo: get_next_line (Projeto 42)

O projeto **get_next_line** tem como objetivo criar uma função em linguagem C que lê uma linha de um ficheiro de cada vez. A função deve ser capaz de lidar com leituras parciais, armazenar o que sobra do buffer entre chamadas e funcionar corretamente com múltiplos descritores de ficheiros.

1. Objetivo da função

A função que vais implementar é: `char *get_next_line(int fd);` Ela deve retornar uma **linha completa** lida do ficheiro descrito por `fd` (file descriptor), incluindo o caractere de nova linha (`\n`) se existir. Se não houver mais nada para ler, deve retornar `NULL`.

2. O papel do BUFFER_SIZE

O **BUFFER_SIZE** é definido no cabeçalho do projeto e indica quantos bytes a função deve ler de cada vez. Como o ficheiro pode não conter uma linha completa dentro de uma única leitura, precisas guardar o conteúdo restante para a próxima chamada da função.

3. Lógica passo a passo

1. 1. Criar uma variável estática para guardar o conteúdo restante entre chamadas.
2. 2. Ler do ficheiro com a função read() até encontrar um '\n' ou o fim do ficheiro.
3. 3. Juntar (concatenar) o que foi lido ao conteúdo anterior.
4. 4. Extrair a linha completa (até ao '\n') para retornar.
5. 5. Guardar o que sobra após o '\n' para uso na próxima chamada.
6. 6. Libertar a memória corretamente quando não houver mais linhas.

4. Funções auxiliares recomendadas

É recomendável criares funções auxiliares para manter o código limpo e modular:

- • ft_strlen() — devolve o tamanho de uma string.
- • ft_strjoin() — junta duas strings (necessário para concatenar as leituras).
- • ft_strchr() — procura um caractere numa string (usado para encontrar '\n').
- • ft_strdup() — duplica uma string.
- • Funções próprias para extrair e guardar a parte restante.

5. Dicas e boas práticas

- • Lembra-te de testar o teu código com diferentes tamanhos de BUFFER_SIZE.
- • Usa verificações rigorosas de erros: read() pode retornar -1.
- • Usa uma variável estática para cada descritor de ficheiro, permitindo múltiplos ficheiros abertos.
- • Verifica se há memory leaks com o comando `valgrind`.
- • Comenta bem o código, especialmente nas funções auxiliares.

6. Exemplo de funcionamento

Supondo que o ficheiro contém o seguinte texto: `Olá Mundo\n Este é um teste\n 42 Lisboa` As chamadas da função devolveriam: 1ª chamada → "Olá Mundo\n"

2ª chamada → "Este é um teste\n"
3ª chamada → "42 Lisboa"
4ª chamada → NULL

O projeto get_next_line é um excelente exercício para entender a manipulação de ficheiros, buffers, gestão de memória e variáveis estáticas em C. É também uma das bases para projetos mais avançados da 42.