03

Memória

Transcrição

[00:00] Eu disse que estou lendo sempre o arquivo inteiro toda vez que quero procurar uma palavra para ser sorteada. E se esse arquivo crescer muito, pode ser que isso seja um problema para a nossa memória. Repare que sempre temos que pensar se o que estou escrevendo vai ser um peso para minha memória, para o processamento do meu computador.

[00:30] A regra geral é que com dados em volume muito pequeno, em que temos um laço simples, sem ser laços intercalados, com um dentro do outro, não costuma ter muito problema de memória. Mesmo que seja um aparelho mobile, não é porque estou usando dez ou dez mil caracteres a mais que minha aplicação vai explodir. Não tem problema nenhum. Seu programa não vai parar de rodar por esses pequenos detalhes. Porem, em algumas situações, por exemplo, um arquivo que tem um giga, não vou querer puxar inteiro para a memória. Posso querer trabalhar com ele de outra maneira.

[01:17] Por exemplo, no momento em que quero escolher a palavra secreta, vou colocar uma variação. O que eu vou fazer é que ao invés de ler todo o conteúdo, meu dicionário vai ter na primeira linha a quantidade de palavras. Isso é, meu dicionário deixa de ser somente essas oito palavras. Ele teria o número oito e aí sim oito palavras. A primeira linha é só o número de palavras que eu tenho dentro do meu arquivo.

[02:05] Vou usar file.new. Isso devolve um arquivo para leitura. Estou lendo esse arquivo. E em algum momento vou querer fechar esse arquivo. Tem diversas maneiras de ler um arquivo. Vimos uma, estamos vendo outra, e tem outras diversas.

[02:28] Eu quero ler uma linha do meu arquivo, só uma. Usamos o gets, que devolve uma linha do arquivo. Eu tenho a quantidade de palavras. Se essa é a quantidade de palavras, qual o número escolhido? É o rand da quantidade de palavras. Agora o que falta é chegar na palavra secreta, certo? Imagine que escolhi o número três. Tenho que ler meu arquivo uma vez, duas vezes, três vezes. Essa última vez é minha palavra secreta. Para isso, posso fazer um for de um até o número escolhido, e uso arquivo.gets. Jogo fora a primeira e a segunda linha e uso a terceira.

[04:15] Recapitulando. Não queríamos levantar o arquivo inteiro para a memória, ter que passar por todo o conteúdo do arquivo. quando quebramos a string em array, temos que passar por todo o conteúdo da string. Depois, quando queremos escolher a palavra qualquer, não precisamos mais. Mas temos que passar por todo o arquivo primeiro para ler ele e depois para encontrar todos os /n. Duas vezes tivemos que passar por todo o arquivo.

[04:44] Aqui não. Aqui logo de cara lemos a quantidade de palavras, sorteamos uma no meio e vamos só até chegar nela. Podemos chutar que em média vai dar o número do meio. Estamos lendo mais ou menos n dividido por dois caracteres, quer dizer, bem mais rápido do que duas vezes n caracteres. É óbvio que se temos cem caracteres, mil caracteres, cem mil caracteres, como usuário final nem vamos sentir diferença de performance.

[05:34] Vamos tentar usar essa função. Quando tentamos jogar, ao invés do escolhe palavra secreta, vai ser escolhe palavra secreta sem consumir muita memória.

[05:47] Quando lemos essas linhas, queremos também tirar a quebra de linha do final. Quando lemos com o gets, fazemos um strip para tirar o /n do final ou o caracteres em branco, tanto do começo quanto do final.

[06:02] No meu dicionário vou colocar o número oito e vou falar para ao invés de chamar escolhe palavra secreta, escolhe palavra secreta sem consumir muita memória. Testando no terminal, funciona.

[06:25] Nós lemos somente até o ponto do arquivo que queríamos, usando file.new, usando gets para ler linha a linha. É só uma das diversas maneiras que temos para ler um arquivo. Tudo depende da maneira como você quer, do que você precisa, para você encontrar a melhor maneira para extrair a informação. No meu caso, mostrei duas maneiras diferentes. Eu vou manter a primeira.