Organização e Recuperação de Dados

Profa. Valéria

3ª Lista de Exercícios

1. Para criar um registro de tamanho fixo em C, podemos usar uma *struct* com campos de tamanho fixo. Como podemos criar registros de tamanho fixo em Python?

   Podemos concatenar os valores dos campos em buffer e preencher o espaço necessário até que o registro tenha o tamanho desejado.

2. Registros de tamanho variável podem ser implementados de várias maneiras, entre elas, utilizando-se (a) registros com um número fixo de campos, (b) colocando uma indicação de tamanho no início do registro (c) utilizando metadados. Qual a vantagem de cada opção?

   (a) tem como vantagem a simplicidade e a possibilidade de se gravar os campos no arquivo à medida que são lidos, por ex., da entrada padrão. No caso de (b), primeiramente é preciso ler todos os campos, concatená-los em alguma estrutura (por ex., um buffer), para só então escrevê-los no arquivo. Dessa forma, (b) aumenta um pouco a complexidade da escrita. Por outro lado, (b) agiliza a leitura, pois permite que um registro seja lido integralmente com uma única instrução de leitura, enquanto em (a) é necessário ler os campos um a um sequencialmente para se identificar onde o registro termina. (c) tem como vantagem tornar o arquivo autodescritivo, uma vez que cada campo é identificado por uma palavra-chave que o descreve. A desvantagem de (c) é que os metadados podem ocupar porção significante do arquivo.

3. Por que a leitura de registros em blocos pode fazer com que o tempo de transferência aumente na busca por um registro?

   Porque mesmo que se deseje ler um único registro do bloco, o bloco terá que ser transferido integralmente para a memória. De maneira similar, a escrita também será feita por blocos, mesmo que apenas um registro do bloco tenha sido modificado.

4. (a) Quantas leituras seriam feitas em média para se encontrar um registro usando busca sequencial em um arquivo contendo 10.000 registros? (b) Se o registro procurado não existe no arquivo, quantas leituras serão feitas? (c) Se o arquivo for organizado em blocos de 20 registros, quantos acessos serão necessários em média?

   (a) 10.000/2 = 5.000 leituras

   (b) 10.000 leituras

   (c) Se os registros estão agrupados em blocos de 20 registros, o arquivo tem fator de bloco k = 20. O caso médio da busca sequencial em blocos é n/2k, então serão necessários 10.000/(2×20) = 250 acessos

5. O que caracteriza o acesso sequencial e o que caracteriza o acesso direto? Ambos os tipos acesso são possíveis em qualquer arquivo?

Temos acesso sequencial quando lemos o arquivo registro a registro, a partir do seu início, para se chegar, por exemplo, a um registro específico.

Temos acesso direto quando sabemos o endereço do registro desejado e podemos acessá-lo diretamente (por meio de um seek), sem necessidade se ler tudo o que veio antes.

Acesso sequencial é possível em qualquer arquivo. Já o acesso direto vai depender de termos acesso aos RRNs (no caso de arquivos com registros de tamanho fixo) ou aos byte-offsets (no caso de arquivos com registros de tamanho variável) dos registros buscados.

6. A figura abaixo representa a saída de um editor hexadecimal (*dump* de arquivo), que descreve os primeiros bytes de um arquivo do mesmo tipo que é produzido pelo programa *escreve_registros.py* (Atividade Prática 1). A última coluna (mais à direita, onde aparecem os pontinhos) foi propositalmente ocultada. Utilize um editor hexadecimal e responda:(a) Qual é o tamanho do 1º registro do arquivo? (b) Qual é o seu conteúdo?

```
00244475    6D707C46    7265647C    38323120    ................
4B6C7567    657C4861    636B6572    7C50417C    ................
36353533    357C0000    00000000    00000000    ................
```

(a) 36 bytes

(b) Dump|Fred|821 Kluge|Hacker|PA|65535|

7. *Suponha* que você tenha um conjunto de campos de tamanho fixo cuja soma dos tamanhos resulta em 30 bytes, ou seja, registros de 30 bytes seriam suficientes para armazená-los. Se esses registros forem ser armazenados em um disco setorizado com setores de 512 bytes, poderia ser interessante utilizar registros de 32 bytes de tamanho em vez de 30. Qual seria a vantagem nesse caso?

Quando se usa registros de tamanho fixo, é interessante que eles tenham um tamanho tal que um número inteiro de registos possa ser armazenado em um setor do disco. No caso do exemplo do exercício, se os registros tivessem 30 bytes, um setor poderia armazenar 17 registros completos e mais uma parte de um registro. Esse registro que ficaria “quebrado” entre dois setores demandaria a leitura de 2 setores para ser recuperado. Por outro lado, se os registros tiverem 32 bytes, cada setor armazenará 16 registros completos, de modo que não haverá registros “quebrados” entre setores. A vantagem então é que, sempre que um setor for lido, ele trará registros completos.

8. Reescreva os programas *escreve_registros.py* e *le_registros.py* (Atividade Prática 1) de modo que eles usem registros de tamanho fixo de 128 bytes.

9. Escreva um programa similar ao programa busca_seq.c (Atividade Prática 2) que implemente as seguintes alterações: (a) em vez ler chaves de busca a partir do teclado, o programa deve lê-las a partir de um arquivo de transações que contém somente as chaves dos registros que devem ser buscados; (b) em vez de imprimir os registros

buscados na tela, os escreva em um arquivo de saída. Considere que tanto o arquivo de dados quanto o arquivo de transação estão desordenados.

(exercícios de programação não serão incluídos no gabarito
por haver mais de uma forma de resolvê-los)