Universidade Federal de São Carlos

RESUMO DE ORI
MATÉRIA DA P1

September 27, 2017

Jonathan Gouvea da Silva
Com base nos slides do Prof. Jander

Contents

1 Representação da Informação

Informação é a atribuição de um significado a alguma coisa (Semântica). **Dado** se refere a um ou mais símbolos. A informação é relacionada com semântica e dado com a sintaxe. Uma informação é representada quando é reduzida a um dado.

Em computação se usa o **byte**, que equivale a 256 símbolos. Temos clássicas representações para inteiros, inteiros com sinal, números reais (float), valores lógicos, caracteres (ASCII).

2 Arquivos em C

2.1 Abertura de arquivo

`FILE *arquivo; arquivo = fopen(nomeDoArquivo, "r+"` O código acima faz a abertura de um arquivo com a função de leitura e escrita. Os modos de abertura são

- `w` - cria um arquivo vazio e permite o uso **apenas** de comandos de escrita.
- `w+` - cria um arquivo vazio e permite escrita e leitura.
- `r` - abre um arquivo apenas com operações de leitura.
- `r+` - abre um arquivo com escrita e leitura, a partir do início do arquivo.
- `a` - abre um arquivo apenas com operações de leitura, acesso feito no final do arquivo.
- `a+` - abre um arquivo com escrita e leitura, acesso no final do arquivo.
- Se adiciona o `b` para identificar que o arquivo é binário.

2.2 Leitura e escrita

As principais funções são, para arquivos binários `fwrite(ponteiro para os elementos a serem escritos, tamanho em bytes de cada elemento, número de elementos, ponteiro para o arquivo)` e `fread(ponteiro para o bloco de memória/variável que a informação será salva, tamanho em bytes de cada elemento, número de elementos, arquivo)`.

Para arquivos de texto temos funções análogas as funções de entrada e saída mais

comuns, como `fprintf`, `fscanf`, `fgets`, `fputc`. Além disso temos funções para busca e etc.

2.3 Fechamento de arquivos

Nunca devemos esquecer de fechar um arquivo! Usamos a função `fclose(arquivo)`.

3 Registros e reuso de espaço

3.1 Representação de campos e registros

Campo é a menor unidade lógica de um arquivo. **Registros** são um conjuntos de dados associadas a mesma pessoa, objeto, etc. Campos podem ser representados com o uso de tamanho fixo, prefixo com tamanho do campo, uso de delimitadores ou rótulos.

- Campos com tamanho fixo são muito fáceis de serem localizados, indicados quando os dados são de tamanho constante ou com pouca mudança. Mas com eles costuma existir um grande desperdício de espaço, valores mais longos que o máximo podem ser perdidos e não é uma solução apropriada quando o comprimento dos dados varia muito.

- Prefixo com tamanho de campo é uma solução em que temos economia de espaço, em que valores muito longos não sofrem perdas, podemos ter campos vazios mas a pesquisa é muito prejudicada.

- Campos com delimitadores ajudam na economia de espaço, podemos ter campos vazios, mas a pesquisa é prejudicada e pode ser difícil escolher um delimitador fora do domínio de dados.

- Campos com rótulos tem o lado positivo de podermos ter campos vazios e o arquivo possui informações de forma mais semântica, clara, "humana". Entretanto a pesquisa é prejudicada e pode ser difícil escolher um delimitador fora do conjunto de dados.

Registros também podem ser representados de diferentes formas, com tamanho fixo, com prefixação de tamanho, com delimitadores ou com índice. O único de diferente do anterior são os índices, os outros funcionam de forma análoga.

Para os registros com índice associado, temos uma grande versatilidade, acesso mais rápido, mas devemos fazer a manutenção em dois arquivos, mantendo a consistência.

Podemos acessar registros atráves de buscas sequenciais, $\Theta(n)$, ou com acesso direto, $\Theta(1)$, se soubermos o endereço preciso do registro.

3.2 Reuso de espaço

Existem dois tipos de remoção, a física (em que o dado é realmente retirado) e a lógica (com uma marcação de que o espaço que esse dado ocupa está disponível e o dado não existe mais). Podemos marcar esse espaço como disponível e podemos rastrear as remoções ocorridas e verificar o espaço disponível. Cada registro removido mantém um ponteiro para o próximo.

Caso os registros tenham tamanho fixo a atualização é feita facilmente, já que todos tem o mesmo tamanho, a substituição por um novo registro pode ocorrer sem problemas. Já se o registro é de tamanho variável dependemos de novas estratégias. O novo registro **deve** caber no espaço que queremos alocá-lo. Podemos pensar em alocar no menor espaço que caiba o novo registro, no primeiro que caiba o novo registro e no maior registro existente. Essas estratégias são, respectivamente, *Best-fit*, *First-fit* e *Worst-fit*.

Em caso de sobra de espaço pode ocorrer a chamada fragmentação, interna quando o espaço livre está dentro de registros de tamanho físico, por exemplo, e externo quando está fora de registros. Para combater a fragmentação primeiro devemos transformar tudo em fragmentação externa e compactar o arquivo ou unir espaços "sobrando" adjacentes.

4 Indexação

Um bloco é a unidade de transferência e de armazenamento de arquivos. A organização de arquivos em blocos no disco é essencial para melhorar o desempenho, para não precisarmos subir um arquivo enorme na memória para qualquer busca.

Usamos chaves para construir um índice, assim conseguimos acessar diretamente no arquivo de dados o que desejamos.

Um índice **agrupado** é aquele quando a ordem dos registros coincide com a ordem das entradas dos índices. O índice **não agrupado** é aquele em que não há relação entre a ordem dos registros e ordem das entradas dos índices.

Índice **esparso** é aquele que contém um menor número de entradas que o número de registros de dados. O índice **denso** é aquele em que temos uma entrada de índice para cada registro.

Um índice **primário** tem a ordenação da chave do arquivo de dados coincidindo com

a ordenação das entradas de índice. Já um índice **secundário** não tem correspondência entre a ordem das chaves nos registros e a ordem das entradas de índice.

Um índice primário em **múltiplos níveis** temos um índice apontando para outro índice, pode ser organizada como esparso ou denso.

Podemos ter um índice com repetição de chave, para solucionar esse problema podemos ter um índice denso com as ocorrências de repetição, podemos manter uma entrada por chave e manter os registros de dados de forma consecutiva ou um índice para lista de chaves primárias.

Para blocos, podemos ter cabeçalhos ou não. Podemos ter registros que ficam parte em um bloco e parte em outro bloco consecutivo ou podemos não usar esses particionamentos.

5 Compactação de dados

5.1 Codificação de Huffman

É um código relativamente simples que leva em conta a frequência de ocorrência de cada símbolo. Para obtermos o código binário devemos seguir os seguintes passos

1. Calcular a frequência de ocorrência de cada símbolo
2. Gerar uma árvore para cada símbolo, com apenas um nó, contendo o símbolo e a probabilidade/frequência.
3. Criar uma lista com as árvores e organizar de forma crescente.
4. Retirar as duas primeiras árvores (as menores) e aglutinar em uma única árvore. No ramo esquerdo fica a primeira árvore, a menor, e no direito fica a segunda, maior que a primeira.
5. Reinserir essa árvore junto a lista de árvores e reordenar.
6. Repetir os passos 4 e 5 até haver apenas uma árvore.
7. Percorrer a árvore final e adicionar em uma tabela de símbolos o símbolo de cada nó da árvore e o código binário associado. O binário associado é 0 a cada ramo esquerdo e 1 a cada ramo direito.

5.2 Codificação LZW

O princípio desse método é substituir sequências de caracteres do conjuntos original por códigos de menor comprimento. O algoritmo para essa codiificação é feita pelos seguintes passos

1. Primeiro se cria um dicionário com cada símbolo do texto.

2. Em seguida, se incia o processamento da cadeia dos caracteres. O programa continua a ler os caracteres até ser formada uma sequência que não está no dicionário.

3. É retornado a última sequência formada por essa cadeia de caracteres, no momento *t-1*.

4. Então é adicionada essa sequência no momento t ao dicionário.

5. Os passos 2, 3 e 4 se repetem até o final do texto.