Conceitos e Fundamentos de Arquivos

Prof. Dr. Lucas C. Ribas

Disciplina: Estrutura de Dados II

Departamento de Ciências de Computação e Estatística

UNIVERSIDADE ESTADUAL PAULISTA
"JÚLIO DE MESQUITA FILHO"

IBILCE / UNESP - CÂMPUS DE SÃO JOSÉ DO RIO PRETO

Sumário

- Introdução
  - Características de armazenamento secundário em disco
  - Estrutura de Arquivos
- História de arquivos
- Operações e fundamentos de arquivos
  - Arquivos físicos e lógicos
  - Operações em C

unesp

Memória Secundária

- Informação mantida em memória secundária:
  - HD e SSD
  - Disquetes
  - Fitas magnéticas
  - CD, DVD e Blu-ray
  - Memória flash: pen-drives, mp3-9 player, cartões
  - Outros?
  - Futuro? bactérias[1], átomos, cristais[2]…

[1]DURLEE, D. Germ of an Idea: Hong Kong Researchers Store Data in Bacteria http://geekbeat.tv/bacteria/
[2]KOVAR, J.F. GE Unveils 500-GB, Holographic Disc Storage Technology. http://www.crn.com/news/storage/217200230/ ge-unveils-500-gb-holographic-disc-storage-technology.htm

Terminologia geral

- **arquivo: u**ma entidade no sistema de arquivos que armazena informações e é identificada por um nome. Pode se referir a documentos individuais ou diretórios que agrupam outros arquivos
- **estrutura de arquivo** (*file structure*): um padrão/organização lógica para se organizar dados num arquivo (incluindo ler, escrever e modificar)
- **algoritmo:** um conjunto finito de regras bem-definidas para a solução de um problema em um numero finito de passos
- **estrutura de dados:** um padrão para organizar dados em um algoritmo ou programa

unesp

Terminologia de acesso a arquivos

- **armazenamento volátil:** armazenamento que perde o conteúdo quando não alimentado por energia.
- **armazenamento não-volátil:** armazenamento que retém o conteúdo quando não alimentado por energia.
- **dados persistentes:** informação que é retida mesmo após a execução de um programa que a cria.

unesp

Velocidade de Acesso

- Discos são lentos! (assim como outros dispositivos para armazenamento secundário)
  - No entanto são muito úteis por combinar baixo custo, alta capacidade de armazenamento e portabilidade
- Quão lentos? tempo de acesso:
  - **HD:** ~ 10 milissegundos (já existem discos com acesso a 8 ms)
  - **RAM:** para acessar a mesma informação em memória principal (RAM) o tempo típico é de ~**70 nanossegundos**, ou 0,00007 milissegundos (já existe tecnologia para acesso em até 10 ns)
  - uma diferença da ordem de 140.000 (em configurações típicas atuais, mas essa diferença pode variar entre 100 mil e 300 mil).

unesp

Discos x Memória Principal

- HD são centenas - e até milhares - de vezes mais lentos que memória RAM!
  - Exemplo:
    - O acesso à RAM equivale a buscar uma informação no índice de um livro que está em suas mãos, em 20 segundos
    - O acesso a disco seria o equivalente a requisitar a busca da mesma informação para uma bibliotecária
    - Comparativamente: usando o cálculo anterior, significa que obter a informação demoraria por volta de 777 horas, ou pouco mais de 32 dias.

unesp

Outras Comparações

- **Custo:**
  - **RAM:** ~ R$ 25,00 / GB (Memória 16GB ~ R$ 400,00)
  - **HD:** ~ R$ 0.29 / GB (HD 1TB ~ R$ 300,00)
- **Volatividade:**
  - **RAM:** volátil
  - **HD:** não-volátil
- **Persistência:**
  - **RAM:** informação é retida enquanto o programa que controla as variáveis estiver sendo executado.
  - **HD:** informação pode ser persistente.

Objetivos em estudar estrutura de arquivos

- Em resumo:
  - acesso a disco é muito caro, isto é, lento!
- Então:
  - o número de acessos ao disco deve ser minimizado
  - a quantidade de informações recuperadas em um acesso deve ser maximizada
- Estruturas de organização de informação em arquivos!

Estrutura de Arquivos

- Meta: minimizar as desvantagens do uso da memória externa
  - Minimizar o tempo de acesso ao dispositivo de armazenamento externo
- De forma independente da tecnologia
  - Tempo de Acesso = nro. de acessos * tempo de 1 acesso
  - deve-se ter cautela para não projetar uma estrutura de arquivo muito dependente da tecnologia atual

unesp

Estrutura de Arquivos

- Estruturas de dados eficientes em memória são muitas vezes inviáveis em disco
- Um dos problemas em se obter uma estrutura de dados adequada para disco é a constante necessidade de alterações em arquivos
- O ideal é evitar sequências de acessos (várias requisições à bibliotecária, no exemplo anterior)
  - Isto é que as informações necessárias possam ser obtidas com **apenas 1 acesso** ao disco
  - Se o ideal não pode ser atingido, deseja-se chegar o mais próximo possível

unesp

Estrutura de Arquivos

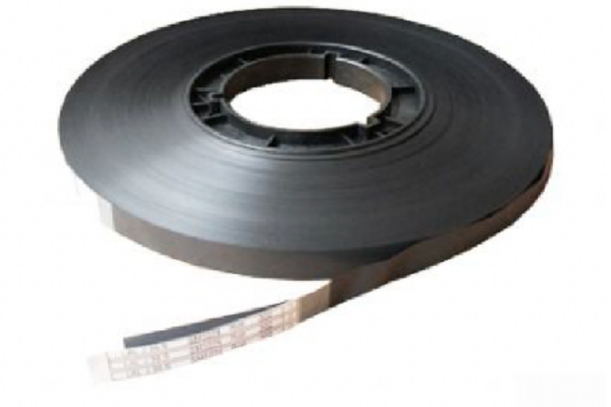
- Exemplo: o método de busca binária permite que um registro pesquisado entre 50.000 seja encontrado em no máximo 16 comparações ($\log_2$ (50.000) ~ 16)
  - Mas acessar o disco 16 vezes para buscar uma informação é tempo demais. Precisamos de estruturas que permitam recuperar esse mesmo registro em dois ou três acessos!
- Queremos estruturas que agrupe informações para permitir recuperar o máximo de informação em uma única operação de acesso
  - por exemplo, ao consultar um pedido de um cliente e buscar suas informações pessoais (nome, endereço, telefone, CPF etc.), é preferível obter todas as informações de uma vez ao invés de procurar em vários lugares

unesp

História

História - Cronologia

- Os primeiros trabalhos com arquivos presumiam o armazenamento em fitas
  - acesso sequencial
  - tamanho dos arquivos cresceu muito e inviabilizou esse tipo de acesso
- São, no entanto, ainda usadas principalmente para armazenamento online redundante pela vida útil longa (até 100 anos)
  - vide caso de recuperação de perdas do Gmail por backup em fitas.



História - Cronologia

- Em discos
  - foram adicionados índices aos arquivos que tornaram possível manter uma lista de chaves e ponteiros para acesso aleatório.
  - mais uma vez o crescimento dos arquivos de índice tornou difícil a sua manutenção.
- Em 1960 o uso de árvores surgiu como solução em potencial, com a desvantagem de crescerem de maneira desigual
  - **Árvores AVL** (1963) foram investigadas para esse problema
  - **Árvores-B:** demoraram 10 anos para serem desenvolvidas pela diferença de abordagem em disco e RAM; crescimento *bottom-up* e acesso sequencial as tornaram a base para os sistemas de arquivos
  - ***Hashing*:** possibilita acesso em tempo constante, mas em arquivos que não se modifiquem
  - ***Hashing* dinâmico:** permitiu modificação no tamanho dos índices e recuperação da informação em no máximo dois acessos.

Operações e Fundamentos de Arquivos

unesp

Arquivos Físicos e Lógicos

- **Arquivo Físico:** um arquivo sempre é físico do ponto de vista do armazenamento. É um conjunto de bytes no disco, geralmente agrupados em setores de dados. Gerenciado pelo sistema operacional

- **Arquivo Lógico:** modo como a linguagem de programação enxerga os dados. Uma sequência de bytes, eventualmente organizados em registros ou outra estrutura lógica

  - Analogia telefone

  - O arquivo lógico pode se relacionar a um arquivo em disco ou a outros dispositivos de E/S

- Assim, a linha de comunicação aberta pelo sistema operacional para o programa é chamado de arquivo lógico, o que é distinto do arquivo físico no disco ou fita, gerenciado apenas pelo sistema operacional

unesp

Arquivos Físicos e Lógicos

- No código fonte, uma instrução liga o arquivo físico a uma variável lógica. Uma vez ligado, é preciso declarar o que desejamos executar no arquivo. Em geral duas opções:
  - abrir um arquivo existente, ou
  - criar um novo arquivo, apagando qualquer conteúdo anterior no arquivo físico.
- Após abrir um arquivo estaremos posicionados no início do arquivo e portanto prontos para escrever ou ler.

unesp

Arquivos Físicos e Lógicos

- Exemplo: associação entre arquivo físico e lógico:
  - Em C na stdio.h:

```
#define FOPEN_MAX (20)
typedef struct _iobuf
{
  char* _ptr;
  int _cnt;
  char* _base;
  int _flag;
  int _file;
  int _charbuf;
  int _bufsiz;
  char* _tmpfname;
} FILE;
```

unesp

Arquivos Físicos e Lógicos

- Abertura de arquivo:

```
FILE *p;
if((p=fopen(fopen("meuarq.dat", "r"))==NULL)
  printf("erro...")
else…
```

- `FILE *p=fopen(<filename>,<flags>)`
- `<filename>`: nome do arquivo a ser aberto
- `<flags>`: modo de abertura
  - **r**: apenas leitura (o arquivo precisa existir)
  - **w**: cria arquivo vazio para escrita (apaga um arquivo já existente)
  - **"a"** adiciona (append) texto no final do arquivo (se arquivo não existe, cria)
  - **r+**: abre arquivo para leitura e escrita
  - **w+**: cria arquivo vazio para leitura e escrita
  - **a+**: abre arquivo para leitura e adição de dados

unesp

Arquivos Físicos e Lógicos

- Fechamento de arquivo: transfere o restante da informação no buffer e desliga a conexão com o arquivo físico

```
fclose(<fd>)
fclose(p)
```

  - `fd`: file descriptor, do tipo ponteiro FILE

- Por que se utiliza *buffer*?

Bufferização

- Toda operação de I/O é ‘bufferizada’
  - Memória Cache: I/O discos 256K, 640K
  - Os bytes passam por uma ‘memória de transferência’ de tamanho fixo e de acesso otimizado, de maneira a serem transferidos em blocos (ex. Microfone)
  - espera para ver se consegue acumular mais bytes indo para o mesmo setor antes de transmitir alguma coisa
- Qual o tamanho dos blocos de leitura/escrita?
  - Depende do SO e da organização do disco
  - Sistema de arquivo: gerencia a manipulação de dados no disco, determinando como arquivos podem ser gravados, alterados, nomeados ou apagados
    - Ex. No Windows, é determinado pela FAT – File Allocation Table (FAT16, FAT32 ou NTFS)

unesp

Leitura e Escrita em C

- Por caractere:

```
<char> = fgetc(<fd>)
fputc(<char>,<fd>)
foef(<fd>)//retorna 1 se fim de arquivo
```

- Por cadeia de caractere:

```
fgets(<char*>,<nro_caracteres>,<fd>)
fputs(<char*>,<fd>)
```

- Por dados formatados:

```
fprintf(<FILE>, ''formatacao'', …)
fscanf(<FILE>, ''formatacao'', …)
```

Referências

- FOLK, M.J. File Structures, Addison-Wesley, 1992.
- File Structures: Theory and Pratice”, P. E. Livadas, Prentice-Hall, 1990;
- Contém material extraído e adaptado das notas de aula dos professores Moacir Ponti, Thiago Pardo, Leandro Cintra e Maria Cristina de Oliveira.