Algoritmos e Estruturas de Dados II

Prof. Ricardo J. G. B. Campello

Fundamentos de Arquivos

Adaptado dos Originais de:

Leandro C. Cintra

Maria Cristina F. de Oliveira

Arquivos

- Informação mantida em memória secundária
  - HD
  - Disquete
  - Fitas magnéticas
  - CD
  - DVD
  - ...

Discos X Memória Principal

- Tempo de Acesso
  - Ordem de grandeza da diferença entre os tempos de acesso pode ser de milhares
  - Em outras palavras, discos (p. ex. HDs) podem ser milhares de vezes mais lentos que memória RAM

3

Discos X Memória Principal

- Capacidade de Armazenamento
  - Disco – alta, a um custo relativamente baixo
  - RAM – limitada pelo custo e espaço
- Tipo de Armazenamento
  - Disco – não volátil
  - RAM – volátil

4

Discos X Memória Principal

- Em resumo:
  - acesso a disco é custoso, isto é, lento!
- Logo:
  - número de acessos ao disco deve ser minimizado
- Para tanto:
  - Estruturas de organização de informação
    - Estruturas de Arquivos (File Structures)

5

Organização de Arquivos

- Meta:
  - minimizar as desvantagens do uso da memória externa
- Objetivo:
  - minimizar o tempo total de acesso ao dispositivo de armazenamento externo
  - de forma independente da tecnologia:

Tempo de Acesso = no. de acessos * tempo de 1 acesso

6

Discos X Memória Principal

- Estruturas de dados eficientes em memória principal são ineficientes em disco
  - Eficiência presume que toda a informação é instantaneamente acessível na memória se o respectivo endereço for conhecido
    - Tempo de acesso desprezível
    - Gargalos são as operações lógicas, aritméticas, ...

7

Disco como Gargalo

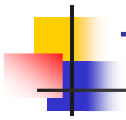
- Muitos processos são "disk-bounded", isto é, CPU e rede têm que esperar pelos dados do disco:
  - Discos são muito mais lentos que CPU e rede

8

Técnicas p/ Minimizar o Problema

- Multi-Programação:
  - CPU trabalha em outro processo enquanto aguarda o disco
- Striping:
  - dados repartidos em vários *drives* (paralelismo)
- RAID (*Redundant Array of Inexpensive Disks*):
  - tecnologia baseada em striping
  - http://linas.org/linux/raid.html

9



Técnicas p/ Minimizar o Problema

- Disk Cache:
  - blocos de memória RAM configurados para conter páginas de dados do disco.
  - cache é verificado primeiro. Se a informação desejada não é encontrada, um acesso ao disco é realizado, e o novo conteúdo é carregado no cache.
- Organização de Arquivos !

10

Arquivo Físico e Arquivo Lógico

- **Arquivo Físico:**
  - seqüência de bytes armazenados no disco
    - armazenamento em geral não é fisicamente seqüencial
- **Arquivo Lógico**:
  - arquivo como visto pelo aplicativo que o acessa
    - freqüentemente visão é seqüencial
- **Associação Arquivos Físico – Lógico**:
  - iniciada pelo aplicativo, gerenciada pelo S.O.

11

Arquivo Físico e Arquivo Lógico

- **Arquivo Físico**:
  - conjunto de bytes no disco
    - geralmente agrupados em setores e clusters de dados
    - gerenciado pelo sistema operacional
- **Arquivo Lógico**:
  - modo como a *linguagem de programação* ou *aplicativo* enxerga os dados
    - seqüência de bytes eventualmente organizados em *registros* ou outra *estrutura lógica*

12

Associação Arquivo Físico – Lógico

- Em Pascal:

  file arq; ou **var** arq: **text**;

  assign(arq, 'meu_arq.dat');

- Em C: (associa e abre – nesse exemplo para escrita)

```
file *pt_arq;
if ( (pt_arq=fopen("meu_arq.dat", "w")) == NULL )
    printf("erro...")   // p. ex. disco cheio ou protegido
else ...
```

13

Revisão Arquivos em C

- pt_arq = **fopen**("filename","flags")
  - filename: nome do arquivo a ser aberto
  - flags: controla o modo de abertura
    - **r**: abre arquivo texto somente para leitura
    - **w**: cria arq. texto para escrita (se já existe, é apagado)
    - **a**: anexa a arquivo texto (append)
    - **r+**: abre arquivo texto para leitura / escrita
    - **w+**: cria arquivo texto para leitura / escrita
    - **a+**: anexa ou cria um arquivo texto para leitura / escrita
    - Notas sobre modo texto:
      - Acesso seqüencial
      - Conversões ASCII ↔ Binário e possíveis conversões CR/LF ↔ LF

14

Revisão Arquivos em C

- pt_arq = **fopen**("filename","flags")
  - filename: nome do arquivo a ser aberto
  - flags: controla o modo de abertura
    - **rb**: abre arquivo binário somente para leitura
    - **wb**: cria arq. binário para escrita (se já existe, é apagado)
    - **ab**: anexa a arquivo binário (append)
    - **r+b**: abre arquivo binário para leitura / escrita
    - **w+b**: cria arquivo binário para leitura / escrita
    - **a+b**: anexa ou cria um arquivo binário para leitura / escrita
    - Notas sobre modo binário:
      - Acesso seqüencial ou direto
      - Nenhuma conversão

15

Fechamento de Arquivos

- Encerra a associação entre arquivos lógico e físico, garantindo que todas as informações sejam atualizadas e salvas
  - descarrega conteúdo dos *buffers* de E/S
  - Em C: **fclose**(pt_arq)
- S.O. fecha o arquivo se o programa não o fizer
  - Previne contra interrupção
  - Libera os recursos associados ao arquivo para outros arquivos

16

Revisão Arquivos em C

- Funções de Leitura / Escrita
  - **fread**()
  - **fwrite**()
    - dados lidos/escritos como registros ou blocos de bytes
    - exemplo: **fwrite**(&v, sizeof(**int**), 10, pt_arq);
    - **modo binário**
  - **fgetc**()
  - **fputc**()
    - dados lidos/escritos um caractere por vez

17

Revisão Arquivos em C

- Funções de leitura / escrita
  - **fgets**()
  - **fputs**()
    - dados lidos/escritos como *strings*
  - **fscanf**(fd)
  - **fprintf**(fd)
    - dados lidos/escritos de modo formatado

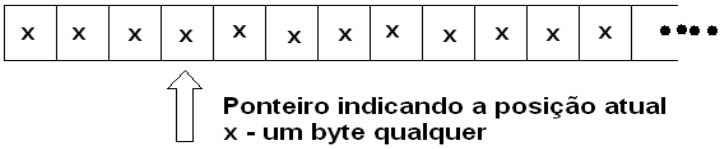
18

Revisão Arquivos em C

- Ponteiro de arquivo:
  - controla o próximo byte a ser lido / escrito
- Controle de final de arquivo:
  - **feof**(pt_arq) – função com retorno lógico
  - Retornos de funções de leitura:
    - **EOF** (tipo especial)
      - **fscanf**, **fgetc**, **fgets**, ...
    - No. de bytes lidos por **fread**

19

O Ponteiro no Arquivo Lógico



20

Acesso Seqüencial vs Aleatório

- Leitura Seqüencial:
  - ponteiro de leitura avança byte a byte (ou por blocos), a partir de uma posição inicial
- Acesso Aleatório (Direto - Seeking):
  - acesso envolve o posicionamento do ponteiro em um byte ou registro arbitrário

21

Revisão Arquivos em C

- Seeking:
  - mover o ponteiro para uma certa posição no arquivo
  - seeking em C
    - bol = **fseek**(pt_arq, byte-offset, origin)
      - bol: retorno lógico (0 ou 1 - mal ou bem sucedido)
      - pt_arq: ponteiro arquivo
      - byte-offset: deslocamento, em bytes, a partir de *origin*
      - origin:
        - **SEEK_SET** (tipo especial – início do arquivo)
        - **SEEK_CUR** (tipo especial – posição atual)
        - **SEEK_END** (tipo especial – fim do arquivo)

22

Bibliografia

- **M. J. Folk and B. Zoellick, *File Structures: A Conceptual Toolkit*, Addison Wesley, 1987.**
- **Schildt, H. "C Completo e Total", 3a. Edição, Pearson, 1997.**

23