Organização em Campos e Registros

Prof. André Backes | @progdescomplicada

Organização de arquivos

- De modo geral, as informações em arquivos são organizadas logicamente em campos e registros
  - campos e registros são conceitos lógicos
  - possuem associação com o arquivo lógico

Produto	Quantidade
Pão	10
Leite	2
Manteiga	1

Organização de arquivos

- Campo
  - Menor unidade lógica de informação em arquivo
  - É uma ferramenta conceitual
  - Não está associado a um conceito físico
- Registro
  - É um conjunto de campos agrupados

Produto	Quantidade
Pão	10
Leite	2
Manteiga	1

Organização de arquivos

- Dependendo de como a informação é mantida, campos lógicos sequer podem ser recuperados
- Exemplo
  - Suponha que desejamos armazenar em um arquivo a descrição e quantidade de vários produtos
  - Suponha que decidimos representar os dados como uma sequência simples de bytes caracteres sem delimitadores, contadores, etc.

Organização de arquivos

- Exemplo

Produto	Quantidade
Pão	10
Leite	2
Manteiga	1

Pão10Leite2Manteiga1

Organização de arquivos

- Uma vez escritas as informações, em princípio não existe como recuperar as unidades lógicas
  - Perde-se a integridade das unidades fundamentais de organização dos dados
  - Essas unidades são agregados de caracteres
  - Não é possível saber onde termina um campo e onde começa outro.

Organização de arquivos em Campos

- Existem várias maneiras de organizar um arquivo mantendo a identidade dos campos
  - A organização anterior não proporciona isso
- Estruturas de Organização de Campos:
  - Com tamanho fixo
  - Indicador de comprimento
  - Uso de delimitadores
  - Uso de *tags*

Campos com tamanho fixo

- Cada campo ocupa um tamanho fixo, pré-estabelecido, no arquivo
  - por exemplo: 4 bytes
- O fato do tamanho ser conhecido garante que é possível recuperar cada campo

```
struct item{
    char produto[20];
    int quantidade;
};
```

P	ã	o																		10
L	e	i	t	e																2
M	a	n	t	e	i	g	a													1

Campos com tamanho fixo

- O espaço alocado (e não usado) aumenta desnecessariamente o tamanho do arquivo
- Desperdício de memória secundária
- Inapropriada quando existe uma grande variabilidade nos tamanhos dos campos
- Razoável apenas se o comprimento dos campos é realmente fixo, ou apresenta pouca variação

```
struct item{
    char produto[20];
    int quantidade;
};
```

P	ã	o																	10
L	e	i	t	e															2
M	a	n	t	e	i	g	a												1

Campos com indicador de comprimento

- O tamanho de cada campo é armazenado imediatamente antes da informação
  - armazenamento binário ou ASCII
  - armazenamento binário: requer um único byte se o tamanho do campo é inferior a 256

Produto	Quantidade
Pão	10
Leite	2
Manteiga	1

[indicador de tamanho][dados do campo]

3Pão210
5Leite12
8Manteiga11

3Pão2105Leite128Manteiga11

Campos com indicador de comprimento

- Vantagem
  - Campos de dados sem espaço vazio
- Desvantagens
  - *overhead* - campos de tamanho precisam ser lidos e escritos
  - Um passo a mais na pesquisa
- Aplicabilidade
  - Situações em que a variabilidade do tamanho dos campos é muito grande

Campos separados por delimitadores

- Um caractere especial (delimitador) é inserido ao final de cada campo
  - O caractere delimitador não pode ser um caractere válido
  - Pode ser um caractere ASCII não imprimível
  - Espaços em branco podem não servir

Produto	Quantidade
Pão	10
Leite	2
Manteiga	1

Pão#10#Leite#2#Manteiga#1

Pão|10|Leite|2|Manteiga|1

Campos separados por delimitadores

- Vantagem
  - Campos de dados sem espaço vazio
- Desvantagens
  - *overhead* - delimitadores precisam ser lidos e escritos
  - Leitura byte a bye
  - Delimitador precisa estar fora do domínio de leitura
- Aplicabilidade
  - Situações em que a variabilidade do tamanho dos campos é muito grande

Campos separados por uso de tags

- Campos são identificados com uma expressão do tipo "keyword=value“
- Um caractere especial (delimitador) é inserido ao final de cada campo

Produto	Quantidade
Pão	10
Leite	2
Manteiga	1

Produto=Pão#Quantidade=10#
Produto=Leite#Quantidade=2 #
Produto=Manteiga#Quantidade=1#

Campos separados por uso de tags

- Vantagens
  - Campo fornece informação semântica sobre si
  - Mais fácil identificar o conteúdo do arquivo, trocar informações
  - Mais fácil identificar campos "vazios"
    - keyword não aparece
- Desvantagens
  - *overhead* - delimitadores e keywords precisam ser lidos e escritos
  - Delimitador precisa estar fora do domínio de leitura
  - As keywords podem ocupar uma porção significativa do arquivo
- Aplicabilidade
  - Situações em que é útil ou necessário armazenar informações semânticas sobre os dados

Organização de arquivos em Registros

- Registro
  - Conjunto de campos agrupados, os quais estão logicamente associados a uma mesma entidade
  - Permite a representação de um arquivo em um nível de organização mais alto
- Assim como os campos, um registro é uma ferramenta conceitual
  - Está associado ao arquivo lógico
  - Não existe, necessariamente, no sentido físico
  - Outro nível de organização imposto aos dados

Produto	Quantidade
Pão	10
Leite	2
Manteiga	1

Organização de arquivos em Registros

- Registros com tamanho fixo
  - Campos de tamanho fixo
  - Campos de tamanho variável
- Registros com tamanho variável
  - Número fixo de campos
  - Uso de delimitadores
  - Indicador de tamanho
  - Uso de índice

Registros com tamanho fixo

- Um dos métodos mais comuns de organização de arquivos
- Assume que todos os registros ocupam o mesmo número de bytes na memória
  - Os campos podem ou não ocupar o mesmo número de bytes (tamanho fixo ou variável)
  - struct em C: registro de tamanho fixo com campo de tamanho fixo

```
struct item{
    char produto[20];
    int quantidade;
};
```

Registros com tamanho fixo

- Um dos métodos mais comuns de organização de arquivos:
  - É simples e permite acesso direto aos registros por RRN (*Relative Record Number*)
- Porém, pode ser inapropriado...
  - Desperdício de memória secundária – fragmentação

Registros com tamanho fixo

- Acesso direto com RRN demanda o uso de registros de tamanho fixo T
  - Posição de início do registro (byte offset):
  - Byte Offset = RRN * T
- Exemplo:
  - Registro na posição lógica 546 a partir do início
  - Tamanho de cada registro é 128
  - Byte offset = 546 x 128 = 69.888 bytes

Memória
Registro 1
Registro 2
Registro 3
Registro 4
Registro 5
...
Registro N

Registros com tamanho fixo

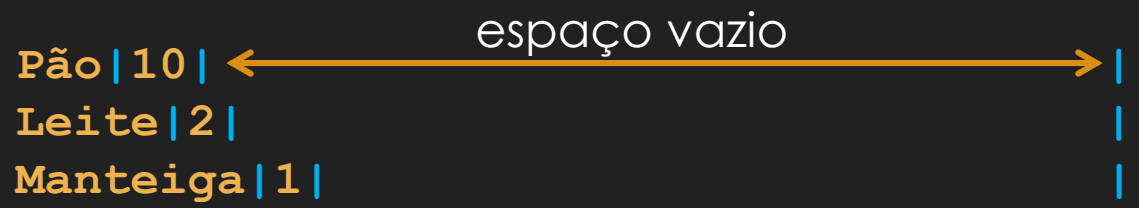
- Operação de seek em um registro de tamanho fixo com campo também de tamanho fixo

```c
#include <stdio.h>
#include <stdlib.h>

struct cadastro{
    char nome[20], rua[20];
    int idade;
};

int main(){
    FILE *f = fopen("arquivo.txt","rb");
    struct cadastro cad;
    if(f == NULL){
        printf("Erro na abertura\n");
        system("pause");
        exit(1);
    }

    fseek(f,2*sizeof(struct cadastro),SEEK_SET);
    fread(&cad,sizeof(struct cadastro),1,f);

    printf("%s\n%s\n%d\n",c.nome,c.rua,c.idade);

    fclose(f);

    return 0;
}
```

Registros com tamanho fixo

- Campo com tamanho variável: acrescenta espaço vazio no final

Produto	Quantidade
Pão	10
Leite	2
Manteiga	1



Registros com número fixo de campos

- Cada registro contém um número fixo de campos
- Tamanho do registro, em bytes, é variável
- Neste caso, os campos são separados por um delimitador
- Pode ser combinada com qualquer estrutura de campo de tamanho variável

Registros com número fixo de campos

- Exemplo:
- Dois campos por registro, com delimitador de campos

Pão#10#Leite#2#Manteiga#1 ...

Pão | 10 | Leite | 2 | Manteiga | 1 ...

Registros com número fixo de campos

- Exemplo:
- Dois campos por registro, com campos com indicador de tamanho

[indicador de tamanho][dados do campo]

3Pão2105Leite128Manteiga11 ...

Registros com número fixo de campos

- Vantagem
  - Campos de dados sem espaço vazio
- Desvantagens
  - *Overhead*
  - No caso dos delimitadores, precisam ser lidos e escritos. Leitura byte a bye. Delimitador precisa estar fora do domínio de leitura
  - No caso dos campos de tamanho, eles também precisam ser lidos e escritos. Isso representa um passo a mais na pesquisa

Registros com indicador de tamanho

- Utiliza um indicador de tamanho na frente da cada registro.
- Esse indicador fornece o tamanho total, em bytes, do registro.
- Os campos são separados internamente por delimitadores.

6Pão|107Leite|210Manteiga|1 ...

Registros com indicador de tamanho

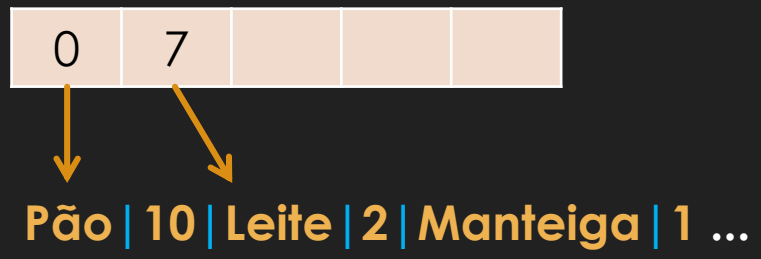
- Vantagem
  - Campos de dados sem espaço vazio
  - Boa solução para registros de tamanho variável
  - Permite acesso direto ao registro seguinte
- Desvantagens
  - *overhead* - campos de tamanho precisam ser lidos e escritos
  - Um passo a mais na pesquisa

Registros com índice para início

- Utiliza um arquivo secundário para manter as informações sobre o endereço do primeiro byte de cada registro
  - os campos são separados por delimitadores
- Indica o deslocamento (*byte offset*) de cada registro relativo ao início do arquivo

Registros com índice para início

- *Byte offset*
  - permite encontrar o começo de cada registro
  - permite calcular tamanho dos registros



Registros com índice para início

- Vantagens
  - Flexibilidade
- Desvantagens
  - Necessidade de acessar 2 arquivos e armazenar conjuntamente
  - Possui um passo a mais na pesquisa, necessidade de manter consistência do índice

Registros com delimitadores de campo e registro

- Separa os registros com delimitadores similares aos de fim de campo
- Delimitador de campos é mantido, sendo que o método agora combina dois delimitadores
  - Outro caractere deve ser utilizado

Pão#10#|Leite#2#|Manteiga#1#|...

Registros com delimitadores de campo e registro

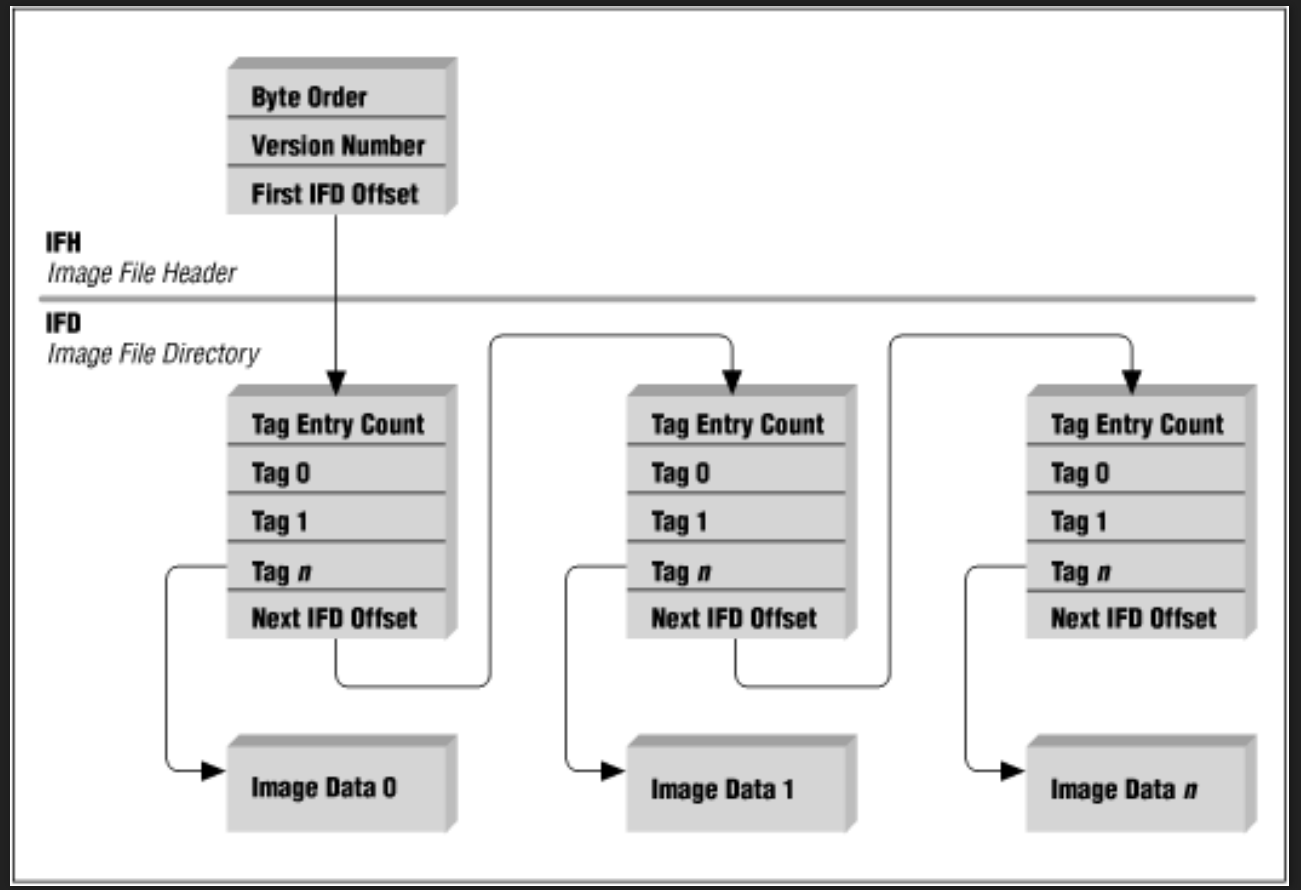
- Vantagem
  - Campos de dados sem espaço vazio
  - Permite número variável de campos
- Desvantagens
  - *overhead* - delimitadores precisam ser lidos e escritos
  - Leitura byte a bye
  - Os dois delimitador precisam estar fora do domínio de leitura

Header records

- Guardam a estrutura do arquivo em um cabeçalho
- Exemplos de informações:
  - Nome dos campos
  - Tamanho de cada campo
  - Número de campos por registro
  - Metadados (exemplo: “propriedades”, data, autor, cantor, etc.)

Header records

- Arquivo TIFF
  - Permite armazenar várias imagens dentro de um mesmo arquivo



Observações

- Nenhum dos métodos descritos é apropriado para todas as situações
- Escolha do método depende da natureza dos dados e da aplicação para que eles serão usados