UnidadeV – Pesquisa de dados Arquivo em Linguagem C

Sumário

4 Operação de Pesquisa e Busca em arquivos

Em C, a busca e pesquisa em arquivos binários geralmente envolve a leitura dos dados do arquivo em uma estrutura na memória, seguida pela análise desses dados para encontrar o que você está procurando.

A pesquisa em arquivos é uma operação fundamental em muitos aplicativos de software e sistemas de banco de dados. Ela permite que os desenvolvedores localizem e recuperem informações específicas armazenadas em arquivos de dados.

A importância da pesquisa em arquivos pode ser resumida da seguinte forma:

1. **Recuperação eficiente de dados:** A pesquisa em arquivos permite que os dados sejam recuperados de forma eficiente com base em critérios específicos, como ID, nome, data etc. Isso é essencial em muitos aplicativos para encontrar e exibir informações relevantes para o usuário.
2. **Organização e gerenciamento de dados:** Ao realizar pesquisas em arquivos, os desenvolvedores podem organizar e gerenciar melhor os dados armazenados. Por exemplo, os registros podem ser ordenados com base em critérios específicos, facilitando a localização rápida de informações relevantes.
3. **Facilidade de manutenção:** A capacidade de pesquisar e localizar rapidamente dados específicos em arquivos facilita a manutenção de aplicativos e sistemas. Se os dados precisarem ser atualizados, removidos ou adicionados, a capacidade de pesquisa torna o processo mais eficiente e preciso.
4. **Melhor experiência do usuário:** A pesquisa eficiente em arquivos leva a uma melhor experiência do usuário, permitindo que os usuários encontrem rapidamente as informações que estão procurando. Isso pode aumentar a satisfação do usuário e a usabilidade do aplicativo ou sistema.
5. **Suporte a recursos avançados:** Em sistemas mais avançados, a pesquisa em arquivos pode ser combinada com recursos como indexação, pesquisa por proximidade, pesquisa por similaridade e pesquisa em grande escala. Esses recursos permitem lidar com grandes volumes de dados e realizar pesquisas complexas em tempo hábil.

4.1 Pesquisa por deslocamento

Vamos analisar o trecho de código abaixo de uma pesquisa por deslocamento:

- a pesquisa por deslocamento é uma abordagem comum para buscar registros em um arquivo binário, especialmente quando os registros têm um tamanho fixo.
- Para realizar a pesquisa por deslocamento, o programa precisa saber onde cada registro está localizado no arquivo. Isso é geralmente feito multiplicando o número de registros desejados pelo tamanho de cada registro. Por exemplo, se cada registro tiver 100 bytes e quisermos acessar o terceiro registro, o deslocamento seria 2 * 100 = 200 bytes. A linguagem C utiliza o tamanho em bytes da estrutura para calcular o deslocamento necessário para alcançar um determinado registro no arquivo. Se cada registro tiver um tamanho fixo, você pode calcular o deslocamento multiplicando o tamanho de um registro pelo número de registros que você quer pular. A linguagem C utiliza indexação baseada em 0, assim o primeiro está indexado em 0, o segundo em 1 e o terceiro em 2.
- Uma vez calculado o deslocamento, o programa move-se o ponteiro do arquivo para essa posição usando a função **fseek()**. Isso permite acessar diretamente o registro desejado no arquivo.
- Após mover o ponteiro do arquivo para a posição correta, o próximo passo é ler o registro desejado. Isso é feito usando a função **fread()**, que lê os dados do arquivo e os armazena em uma estrutura na memória. Uma vez que o registro é lido, ele pode ser processado conforme necessário. Por exemplo, você pode imprimir os dados na tela ou executar outras operações com base nos valores lidos.

```
cout<<"\n Digite o id do registro:";
fflush(stdin);
cin>>idcontato;
deslocamento = idcontato * sizeof(cad);
    if (fseek(fp,deslocamento, 0) != 0) {
        cout<<"\nImpossivel localizar o registro";
    }
    else{
        fread(&cad,sizeof(cad),1,fp);
        cout<<"Nome: ";
        cout<<cad.nome<<endl;
        cout<<"Ano: ";
        cout<<cad.ano<<endl;
        fread(&cad, sizeof(cad),1,fp);
        system("PAUSE");
        system("cls");
    }
```

if ((fp = fopen("arqcad.dat","rb")) == NULL){...}: Tenta abrir o arquivo "arqcad.dat" para leitura binária. Se falhar, exibe uma mensagem de erro.

- Se o arquivo for aberto com sucesso, o código dentro do bloco **else** é executado.

idcontato=> Lê o ID do registro a ser localizado.

deslocamento = idcontato * sizeof(cad)=> Calcula o deslocamento no arquivo com base no tamanho do registro.

if (fseek(fp,deslocamento, 0) != 0) {...}: Move o ponteiro do arquivo para o deslocamento calculado. Se falhar, exibe uma mensagem de erro.

- Se o deslocamento for bem-sucedido, o código dentro do bloco **else** é executado.

fread(&cad,sizeof(cad),1,fp) =>Lê os dados do registro no arquivo e armazena-os na estrutura cad.

cout<<"Nome: "; cout<<cad.nome<<endl;: Exibe o nome do registro.

cout<<"Ano: "; cout<<cad.ano<<endl;: Exibe o ano do registro.

4.3 Pesquisa por comparação

Vamos analisar o trecho de código abaixo de uma pesquisa por comparação:

- A pesquisa por comparação é uma abordagem de pesquisa por comparação, onde você procura por registros que correspondam a critérios específicos, como um ID ou um nome. E começa após abrir o arquivo, na sequência deve ler cada registro do arquivo em um loop. Para cada registro lido, você compara o campo relevante (como ID ou nome) com o valor que você está procurando.
- Se um registro correspondente for encontrado, pode-se fazer o que desejar com esse registro, como exibi-lo na tela ou armazená-lo em uma estrutura de dados para uso posterior. Se nenhum registro correspondente for encontrado, deve-se informar ao usuário que o registro não foi encontrado.

```
fflush(stdin);
cout<<"\n Digite o nome: ";
fflush(stdin);
gets(aux);
system("cls");
    while (!feof(fp)){
        if (strcmp(cad.nome, aux) == 0){
            cout<<"Nome: ";
            cout<<cad.nome<<endl;
            cout<<"Ano: ";
            cout<<cad.ano<<endl;
            system("PAUSE");
            system("cls");
        }
    fread(&cad, sizeof(cad), 1, fp);
    }
```

gets(aux);: Lê o nome digitado pelo usuário e o armazena na variável **aux**. É o dado do registro que se deseja pesquisar.

while (!feof(fp)){...}: Entra em um loop que continua até que o final do arquivo seja alcançado. O operador **!feof(fp)** verifica se não estamos no final do arquivo, enquanto procura o nome (registro) que

seja igual ao conteúdo da varável aux, ou seja, procura o dado até o fim do arquivo ou até encontrar o que se deseja.

if (strcmp(cad.nome, aux) == 0){...}: Verifica se o nome do registro lido do arquivo é igual ao nome digitado pelo usuário. A função **strcmp()** compara duas strings e retorna zero se forem iguais.

A abordagem usando **feof()** pode levar a alguns problemas, especialmente porque a leitura do último registro pode ser repetida, porém é simples e permite encontrar o dado desejado. Uma abordagem mais robusta seria verificar o retorno de **fread()** para determinar se a leitura foi bem-sucedida, em vez de depender de **feof()**.

Em resumo, a pesquisa em arquivos desempenha um papel crucial em muitos aspectos do desenvolvimento de software e sistemas de informação. Ela permite a recuperação eficiente de dados, organização e gerenciamento eficazes, manutenção simplificada, melhor experiência do usuário e suporte a recursos avançados de pesquisa.