Aula 7: Arquivos

Adriano Veloso

Algoritmos e Estruturas de Dados I - DCC/UFMG

Arquivos

Arquivo é um recurso que o sistema operacional promove e que as linguagens de programação tem acesso. É uma abstração que simplifica a criação e manipulação de dispositivos de entrada e saída de dados, não apenas restringindo a arquivos físicos gravados no disco rígido, mas também dispositivos como o teclado e o monitor.

Arquivos no SO

Quando criamos um arquivo, seja através de uma linguagem de programação ou através da interface de usuário, na verdade não estamos propriamente criando o arquivo, mas sim delegando esta tarefa ao Sistema Operacional (Linux, Windows, Mac), que se encarrega de fazer toda a verificação de permissões do usuário que está executando esta tarefa e atribuindo às descrições do arquivo informações como: usuário dono, data de criação, tipo, tamanho, data de acesso, modificação, dentre outras.

Tipos de arquivos

Os arquivos são divididos em duas subcategorias:

1. Binários - Arquivos interpretados apenas pelo SO. Contém apenas 0's e 1's. Programas compilados e bibliotecas são exemplos de arquivos binários;
2. Textos - São arquivos que contém informações, não apenas se restringindo a texto propriamente dito, mas também a arquivos de mídia por exemplo.

Operações básicas

O processo de utilização de um arquivo envolve, no mínimo, três etapas:

1. criação ou abertura do arquivo
2. gravação ou leitura de dados no arquivo
3. fechamento do arquivo.

Criação de arquivos

Na primeira etapa, se o arquivo ainda não existir, ele deve ser criado. Caso o arquivo já existir (pelo fato de ter sido criado em uma execução anterior do programa) ele pode ser aberto para que novos dados sejam acrescentados ou para que os dados guardados nele possam ser lidos.

Leitura ou escrita de dados

A segunda etapa é a que efetivamente lê ou grava dados no arquivo. A linguagem C oferece funções específicas de leitura e de escrita de dados em um arquivo. Se o arquivo for do tipo texto (como este que você está lendo), você deve utilizar funções específicas para arquivos-texto. Se ele for binário (do tipo que armazena registros ou estruturas), você deve utilizar as funções de leitura e/ou escrita em arquivos binários.

Fechamento

A terceira etapa consiste em fechar o arquivo, para que seus dados sejam efetivamente gravados e fiquem protegidos até que o arquivo seja aberto novamente.

Arquivos em C

Na linguagem C, os arquivos não podem ser manipulados diretamente. Eles precisam representados (referenciados, na verdade) por uma variável do tipo FILE*. Logo, sempre que você for utilizar um arquivo crie uma variável do tipo FILE* para poder utilizá-lo. Todas as funções de manipulação de arquivo necessitam de uma variável deste tipo para poder manipular esse arquivo.
Declarar uma variável deste tipo é fácil.
FILE* arquivo;

Arquivos em C

Na linguagem C, os arquivos não podem ser manipulados diretamente. Eles precisam representados (referenciados, na verdade) por uma variável do tipo FILE*. Logo, sempre que você for utilizar um arquivo crie uma variável do tipo FILE* para poder utilizá-lo. Todas as funções de manipulação de arquivo necessitam de uma variável deste tipo para poder manipular esse arquivo.
Declarar uma variável deste tipo é fácil.
FILE* arquivo;
É disponibilizada uma série de funções para trabalhar com este conceito, cujos protótipos estão reunidos em stdio.h.

Arquivos em C

Depois de ter declarado uma variável que vai representar o arquivo, você deve efetivamente tentar abrir ou criar o arquivo. Isso é feito com a função fopen. Porém, antes de usar a função fopen, você deve decidir se vai abrir ou criar o arquivo (isso vai depender de ele já existir ou não na memória secundária) e também deve decidir se ele vai ser um arquivo do tipo texto ou do tipo binário.
Dependendo da sua escolha, a fopen vai ser chamada de uma forma ou de outra, com parâmetros diferentes que indicam as escolhas que você tomou.

Função fopen

variável = fopen("nome_arquivo.extensão", "modo+tipo");
Modos possíveis:

- r → abrir
- w → criar/sobrescrever

Tipos possíveis:

- t → texto
- b → binário

Logo, para abrir um arquivo o modo deve ser "r", e para criar um arquivo (ou sobrescrever um já existente) o modo deve ser "w". Tome muito cuidado com o modo "w", pois todos os dados de um arquivo são apagados, caso ele já exista.

Função fopen

Exemplos:

- Abrindo um arquivo do tipo texto
  arquivo = fopen("teste.txt", "r+t");
- Criando um arquivo do tipo texto
  arquivo = fopen("teste.txt", "w+t");
- Abrindo um arquivo do tipo binário
  arquivo = fopen("teste.txt", "r+b");

Função fopen

Você deve testar se o arquivo foi realmente aberto ou criado. Isso porque as demais funções de manipulação de arquivo não funcionarão se a função fopen não tiver funcionado. Para saber se a função fopen realmente criou ou abriu o arquivo que você solicitou, teste se a variável que representa o arquivo (a variável arquivo, no exemplo) possui algum valor dentro dela. Se ela não possuir um valor (se ele for NULL) é porque o arquivo não foi aberto ou criado:

```
if(arquivo!=NULL)
{
    coloque as funções de manipulação de arquivo aqui dentro
}
```

Funções fprintf e fscanf

Depois de ter aberto ou criado um arquivo, e ter uma variável que o represente dentro do programa, você pode começar a gravar ou a ler dados neste arquivo.

Funções fprintf e fscanf

Depois de ter aberto ou criado um arquivo, e ter uma variável que o represente dentro do programa, você pode começar a gravar ou a ler dados neste arquivo.
Para gravar, utilize a função fprintf, que é muito parecida com a função printf. A única diferença é que há um parâmetro (o primeiro) indicando qual é o arquivo onde os dados serão gravados (ou melhor, impressos).

Funções fprintf e fscanf

Depois de ter aberto ou criado um arquivo, e ter uma variável que o represente dentro do programa, você pode começar a gravar ou a ler dados neste arquivo.
Para gravar, utilize a função fprintf, que é muito parecida com a função printf. A única diferença é que há um parâmetro (o primeiro) indicando qual é o arquivo onde os dados serão gravados (ou melhor, impressos).
Para ler, utilize a função fscanf, que é muito parecida com a função scanf. A única diferença é que há um parâmetro (o primeiro) indicando qual é o arquivo de onde os dados serão lidos.

Funções fprintf e fscanf

Quando você lê ou grava alguma coisa do arquivo, automaticamente você é colocado na próxima posição do mesmo
Esta é a posição de onde será lido ou escrito o próximo dado.
Normalmente, num acesso seqüencial a um arquivo, não temos que mexer nesta posição pois quando lemos um dado a posição no arquivo é automaticamente atualizada.

Função feof

O marcador de fim de arquivo é denominado "EOF". Não se pode ler nada após o "EOF".
O comando feof(arquivo) retorna verdadeiro quando a posição do arquivo atinge o "EOF". Dessa forma, podemos ler todo o conteúdo de um arquivo usando o comando fscanf dentro de um loop while(!feof(arquivo)).

Função feof

O marcador de fim de arquivo é denominado "EOF". Não se pode ler nada após o "EOF".
O comando feof(arquivo) retorna verdadeiro quando a posição do arquivo atinge o "EOF". Dessa forma, podemos ler todo o conteúdo de um arquivo usando o comando fscanf dentro de um loop while(!feof(arquivo)).

```
...
fscanf(arquivo, ...);
while(!feof(arquivo)) {
    ...
    fscanf(arquivo, ...);
    ...
}
...
```

Função fclose

Quando acabamos de usar um arquivo que abrimos, devemos fechá-lo. Para fechar um arquivo basta chamar a função fclose:
fclose(arquivo);
O ponteiro "arquivo" determina o arquivo a ser fechado. A função retorna zero no caso de sucesso.

Função fclose

Quando acabamos de usar um arquivo que abrimos, devemos fechá-lo. Para fechar um arquivo basta chamar a função fclose:
fclose(arquivo);
O ponteiro "arquivo" determina o arquivo a ser fechado. A função retorna zero no caso de sucesso.
Fechar um arquivo faz com que qualquer dado que tenha permanecido no "buffer"associado ao fluxo de saída seja gravado.

Função fclose

Quando acabamos de usar um arquivo que abrimos, devemos fechá-lo. Para fechar um arquivo basta chamar a função fclose:
fclose(arquivo);
O ponteiro "arquivo" determina o arquivo a ser fechado. A função retorna zero no caso de sucesso.
Fechar um arquivo faz com que qualquer dado que tenha permanecido no "buffer"associado ao fluxo de saída seja gravado.
A função exit() fecha todos os arquivos que um programa tiver aberto.

Contato

adrianov@dcc.ufmg.br