

Arquivos – Uma visão em C++



Arquivos – Uma visão em C++

- O que são arquivos?
- Manipulação de arquivos – Com funciona?
- O que são as classes `iostream` e `fstream.h`?
- Exemplo de gravação - (`ofstream`).
- Exemplo de leitura - (`ifstream`)
- Exemplo de gravação e leitura - (`fstream`).





O que são 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.
- 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, estamos 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...



O que são arquivos?



- Todo Sistema Operacional, trabalha em cima de um Sistema de Arquivos, que por sua vez é um dos principais responsáveis na criação e manipulação dos arquivos em dispositivos físicos.
- Cada partição pode receber um Sistema de Arquivos, dentre os mais conhecidos estão: FAT 16, FAT 32, NTFS (Usados pelo Windows), Ext2, Ext3, ReiserFS, XFS(Usados pelo Linux e por outros Unix-like).
- Os arquivos são divididos em duas subcategorias.
 - Binários – Arquivos interpretados apenas pelo SO. Contém apenas 0's e 1's. Programas compilados e bibliotecas são exemplos de arquivos binários.
 - Textos – São arquivos que contém informações, não apenas se restringindo a texto propriamente dito, mas também a arquivos de mídia por exemplo.



O que são arquivos?



- Alguns Sistemas Operacionais, como o Linux, tratam todo o sistema como uma grande gama de arquivos; processos, diretórios, usuários e dispositivos todos são representados por arquivos, e cabe ao SO decidir que tipo é cada arquivo, isto é feito através da leitura do cabeçalho dos mesmos, assim, ao verificar através desta leitura, se o arquivo em questão é uma imagem por exemplo, o SO permite que um programa de imagens "abra" este arquivo e não permite que um programa de reprodução de música seja ativado, pois a estrutura de um arquivo que seja uma imagem é totalmente diferente de um arquivo de audio. Observe que, a verificação de que tipo é o arquivo se dá na leitura do cabeçalho do mesmo, e não na extensão que este possui. Desta forma, podemos encontrar arquivos sem extensão no Linux, o que não impede sua manipulação. No Windows, o SO reconhece o tipo de arquivo a partir da extensão(.txt, .jpg, .exe, ...) do mesmo.

Manipulação de arquivos – Como funciona?

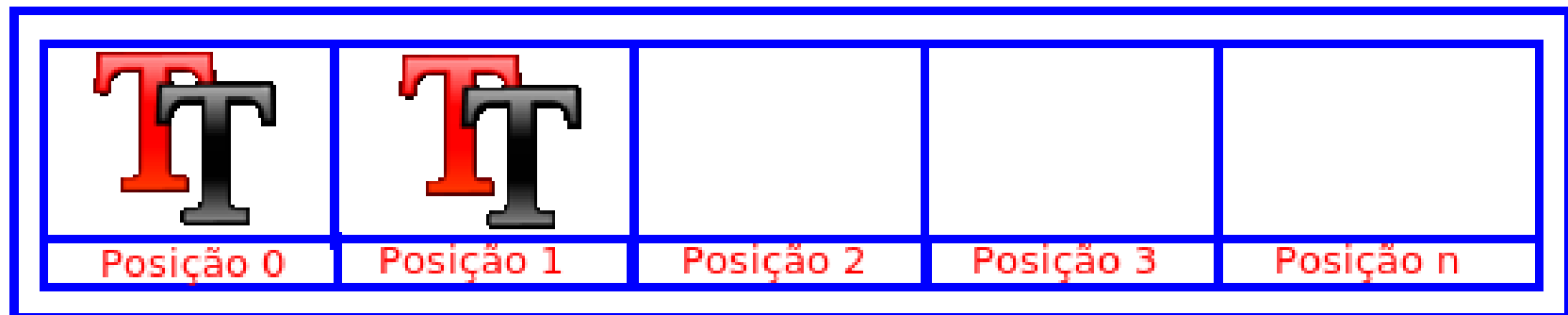


- Para a manipulação (leitura e/ou escrita) e criação de arquivo, é necessário que façamos uma referência do mesmo à um espaço da memória. Para tanto, o C++, utiliza as classes `iostream` e `fstream.h`.
- Para realizar gravação em arquivos, fazemos:
`ofstream fgrava("livro.txt");` //O objeto `fgrava` faz referência ao arquivo `livro.txt`.
- Para realizar leitura em arquivos, fazemos:
`ifstream fleitura("livro.txt");` //O objeto `fleitura` faz referência ao arquivo `livro.txt`.
- Para realizar leitura e gravação em arquivos, fazemos:
`fstream fio;`
`fio.open("livro.txt");` //O objeto '`fio`' recebe a referência ao arquivo `livro.txt`.

Manipulação de arquivos – Como funciona?



- Manipulação de arquivos, são, de certa forma, parecida com a manipulação de vetores, ambos trabalham com o conceito de posição; a diferença se dá no fato de que, arquivos não tem um limite de posições, ao contrário do que ocorre com vetores, assim, teoricamente, podemos adicionar novos registros até o limite físico do disco rígido.



Manipulação de arquivos – Como funciona?



- Podemos escolher a posição em que queremos trabalhar no arquivo, para tanto, usamos a função:

```
<nome_da_objeto_de_entrada_e_saida_de_dados>.seekg(X);
```

- X deve receber a posição do registro no arquivo. Em C++, ao contrário do que ocorre em Algoritmo, X não recebe 0,1,2,3 ..., mas sim a posição do registro em forma binária. Para tanto, usamos o seguinte artifício:

```
sizeof(nome_do_objeto_molde_do_registro)
```

- Posteriormente iremos ilustrar como utilizar a função sizeof para o posicionamento do arquivo.

O que são as classes `iostream` e `fstream.h`?



- O C++, como exemplificado anteriormente, utiliza as classes `ofstream` e `ifstream` (membros da classe `iostream`) para realizar, respectivamente, a gravação e leitura de arquivos. Para realizar ambos em um mesmo arquivo, utilizamos a classe `fstream.h`
- Quando utilizamos `fstream`, necessariamente também precisamos fazer uso da função `open()`, que é a responsável por fazer a referência de um objeto (alocado na memória) com um arquivo. Utilizando `ofstream` e `ifstream`, esta "ligação" com um arquivo é feita diretamente na declaração do objeto.
- Para utilizarmos as classes `iostream` e `fstream`, é necessário que façamos a referência das mesmas no código:

```
#include <fstream.h>  
#include <iostream>
```

O que são as classes `iostream` e `fstream.h`?



- Para fazer a gravação no arquivo, usamos a função:
`<objeto_de_dados>.write((char *)&<objeto_para_ser_gravado>, sizeof(<classe_molde_do_objeto_a_ser_gravado>))`
- Para fazer a leitura no arquivo, usamos a função:
`<objeto_de_dados>.read((char *)&<objeto_para_ser_gravado>, sizeof(<classe_molde_do_objeto_a_ser_gravado>))`

Exemplo de gravação – (ofstream) – Gravando um registro



```
#include <fstream.h>

#include <cstdlib>

#include <stdio.h>

#include <iostream>

using namespace std;

class alunos{

    private:

        int ra;

        char nome[50];

    public:

        alunos();

        void cadastra();

};

alunos :: alunos(){

    ra = 1;

    strcpy(nome,"");

}
```

```
void alunos :: cadastra(){

    cout << "Digite o RA: " << endl;

    cin >> ra;

    cout << "Digite o nome: "<< endl;

    cin >> nome;

}

int main(){

    //Aqui criamos o objeto 'fgrava', que faz referência ao objeto aluno.txt

    ofstream fgrava("alunos.txt");

    //Aqui criamos o objeto 'cad1' usando a classe 'alunos'.

    alunos cad1;

    cad1.cadastra();//Lemos os dados com a função 'cadastra()'.

    fgrava.write( (char *)&cad1, sizeof(alunos));//Aqui gravamos o

    conteúdo do objeto 'cad1' em 'alunos.txt'.

    //Usamos 'sizeof(alunos)' neste caso, para informar o tamanho do registro e

    posteriormente grava -lo no arquivo.

}
```

Resultados - compilação



Resultado esperado do código acima:

Digite o RA:

<Aqui o usuário digita o RA>

Digite o nome:

<Aqui o usuário digita o Nome>

Exemplo de gravação – (ifstream) – Lendo um registro



```
#include <fstream.h>

#include <cstdlib>

#include <stdio.h>

#include <iostream>

using namespace std;

class alunos{
    private:
        int ra;
        char nome[50];
    public:
        alunos();
        void display();
};

alunos :: alunos(){
    ra = 1;
    strcpy(nome,"");
}

void alunos :: display(){
    cout << "Número do RA: " << endl;
    cout << ra << endl;
    cout << "Nome do aluno: " << endl;
    cout << nome << endl;
}

int main(){
    //Aqui criamos o objeto 'fread', que faz referência ao objeto aluno.txt
    ifstream fread("alunos.txt");

    //Aqui criamos o objeto 'cad1' usando a classe 'alunos'.
    alunos cad1;

    fread.read( (char *)&cad1, sizeof(alunos)); //Aqui "lemos" o conteúdo
do arquivo 'alunos.txt' em 'cad1'.
    //Usamos o sizeof(alunos) para ler o registro no tamanho certo da classe
alunos.
    cad1.display(); //Imprime na tela o registro.
}
}
```

Resultados - compilação



Resultado esperado do código acima:

Número do RA:

316938

Nome do aluno:

Matheus

Exemplo de gravação – (ofstream) – Gravando vários registros



```
#include <fstream.h>

#include <cstdlib>

#include <stdio.h>

#include <iostream>

using namespace std;

class alunos{

private:

    int ra;

    char nome[50];

public:

    alunos();

    void cadastra();

};

alunos :: alunos(){

    ra = 1;

    strcpy(nome,"");

}

void alunos :: cadastra(){

    cout << "Digite o RA: " << endl;

    cin >> ra;

    cout << "Digite o nome: "<< endl;

    cin >> nome;

}

int main(){

    int i;

    //Aqui criamos o objeto 'fgrava', que faz referência ao objeto aluno.txt

    ofstream fgrava("alunos.txt");

    //Aqui criamos o objeto 'cad1' usando a classe 'alunos'.

    alunos cad1;

    cout << "Deseja cadastrar um aluno? Tecle 0(zero) para sair e 1 para

confirmar." << endl;

    cin >> i;

    while(i!=0){

        cad1.cadastra();//Lemos os dados com a função 'cadastra()'.

        fgrava.write( (char *)&cad1, sizeof(alunos));//Aqui gravamos o

conteúdo do objeto 'cad1' em 'alunos.txt'.

        //Usamos 'sizeof(alunos)' neste caso, para informar o tamanho do registro e

posteriormente grava -lo no arquivo.

        cout << "Deseja cadastrar um aluno? Tecle 0(zero) para sair e

1 para confirmar." << endl;

        cin >> i;

    }

}
```

Resultados - compilação



Resultado esperado do código acima:

Deseja cadastrar um aluno? Tecele 0(zero) para sair e 1 para confirmar.

1

Digite o RA:

23445

Digite o nome:

Lais

Deseja cadastrar um aluno? Tecele 0(zero) para sair e 1 para confirmar.

1

Digite o RA:

767553

Digite o nome:

Joao

Deseja cadastrar um aluno? Tecele 0(zero) para sair e 1 para confirmar.

0

Exemplo de gravação – (ifstream) – Lendo vários registros.



```
#include <fstream.h>

#include <cstdlib>

#include <stdio.h>

#include <iostream>

using namespace std;

class alunos{
    private:
        int ra;
        char nome[50];
    public:
        alunos();
        void display();
};

alunos :: alunos(){
    ra = 1;
    strcpy(nome,"");
}

void alunos :: display(){
    cout << "Número do RA: " << endl;
    cout << ra << endl;
    cout << "Nome do aluno: " << endl;
    cout << nome << endl;
}

int main(){
    //Aqui criamos o objeto 'fread', que faz referência ao objeto aluno.txt
    ifstream fread("alunos.txt");

    //Aqui criamos o objeto 'cad1' usando a classe 'alunos'.
    alunos cad1;

    while(fread){//enquanto não for F.D.A
        //Aqui "lemos" o conteúdo do arquivo 'alunos.txt' em 'cad1'.
        fread.read( (char *)&cad1, sizeof(alunos));
        //Usamos o sizeof(alunos) para ler o registro no tamanho certo da classe
        alunos.
        cad1.display();//Imprime na tela o registro.
    }
}
```

Resultados - compilação



Resultado esperado do código acima:

Número do RA:

23445

Nome do aluno:

Lais

Número do RA:

767553

Nome do aluno:

Joao

Número do RA:

767553

Nome do aluno:

Joao

Exemplo de gravação e leitura - (fstream).



```
#include <fstream.h>

#include <cstdlib>

#include <stdio.h>

#include <iostream>

using namespace std;

class alunos{
    private:
        int ra;
        char nome[50];
    public:
        alunos();
        void cadastra();
        void display();
};

alunos :: alunos(){
    ra = 1;
    strcpy(nome,"");
}

void alunos :: cadastra(){
    cout << "Digite o RA: " << endl;
    cin >> ra;
    cout << "Digite o nome: "<< endl;
    cin >> nome;
}

void alunos :: display(){
    cout << "Número do RA: " << endl;
    cout<< ra << endl;
    cout << "Nome do aluno: "<< endl;
    cout << nome << endl;
}

int main(){
    int i;

    fstream iodados;
    iodados.open("alunos.txt", ios::ate | ios::out | ios::in);

    alunos cad1;

    cout << "Deseja cadastrar um aluno? Tecla 0(zero) para sair e 1 para
confirmar." << endl;
    cin >> i;
```

Exemplo de gravação e leitura - (fstream).



```
while(i!=0){  
    cad1.cadastra();//Lemos os dados com a função 'cadastra()'.  
    iodados.write( (char *)&cad1, sizeof(alunos));//Aqui gravamos o  
    conteúdo do objeto 'cad1' em 'alunos.txt'.  
    //Usamos 'sizeof(alunos)' neste caso, para informar o tamanho do registro e  
    posteriormente grava -lo no arquivo.  
    cout << "Deseja cadastrar um aluno? Tecele 0(zero) para sair e  
    1 para confirmar." << endl;  
    cin >> i;  
    }  
  
    iodados.seekg(0);  
  
    while (iodados.read( (char *)&cad1, sizeof(alunos)))  
  
        cad1.display();//Imprime na tela o registro.  
  
}
```

Obs.: Em **iodados.seekg(0)**, podemos substituir por **iodados.seekg(sizeof(alunos))**. Se fizermos **iodados.seekg(sizeof(alunos) + sizeof(alunos))**, colocamos o arquivo na segunda posição.

No código acima, foi necessário colocar **iodados.seekg(0)**, pois, depois que adicionamos registros ao arquivo, este chega ao estado de Fim de Arquivo(End of File), colocando 0(zero), voltamos ao começo do arquivo e depois mostramos na tela os registros armazenados no mesmo.

Resultados - compilação



Resultado esperado do código acima[1]:

Deseja cadastrar um aluno? Tecele 0(zero) para sair e 1 para confirmar.

0

Número do RA:

23445

Nome do aluno:

Lais

Número do RA:

767553

Nome do aluno:

Joao

Resultados - compilação



Resultado esperado do código acima[2]:

Deseja cadastrar um aluno? Tecele 0(zero) para sair e 1 para confirmar.

1

Digite o RA:

316938

Digite o nome:

Matheus

Deseja cadastrar um aluno? Tecele 0(zero) para sair e 1 para confirmar.

1

Digite o RA:

65734

Digite o nome:

Mary

Deseja cadastrar um aluno? Tecele 0(zero) para sair e 1 para confirmar.

0

Resultados - compilação



Número do RA:

23445

Nome do aluno:

Lais

Número do RA:

767553

Nome do aluno:

Joao

Número do RA:

316938

Nome do aluno:

Matheus

Número do RA:

65734

Nome do aluno:

Mary

Exemplo de gravação e leitura - (fstream).



Em `iodados.open("alunos.txt", ios::ate | ios::out | ios::in);`, o segundo argumento indica o modo de abertura do arquivo. Abaixo, temos as opções possíveis para abertura(O pipe é usado para combinar as opções):

Modos de abertura	Descrição
<code>ios::in</code>	Abre para leitura (default de ifstream).
<code>ios::out</code>	Abre para gravação (default de ofstream),
<code>ios::ate</code>	Abre e posiciona no final do arquivo.(Este modo trabalha
<code>com leitura e gravação)</code>	
<code>ios::app</code>	Grava a partir do fim do arquivo
<code>ios::trunc</code>	Abre e apaga todo o conteúdo do arquivo
<code>ios::nocreate</code>	Erro de abertura se o arquivo não existe
<code>ios::noreplace</code>	Erro de abertura se o arquivo existir
<code>ios::binary</code>	Abre em binário (default é texto)

É recomendada a utilização da função `<nome_do_objeto_de_dados>.close`, para fechar o arquivo, depois de manipula-lo, no final do código. Alguns compiladores podem não aceitar e gerar erros se o arquivo não é fechado.

Fontes



- www.dimap.ufrn.br/~adilson/DIm327/arquivos.pdf
- <http://www.vivaolinux.com.br/artigo/Fundamentos-do-sistema-Linux-arquivos-e-diretorios?pagina=2>