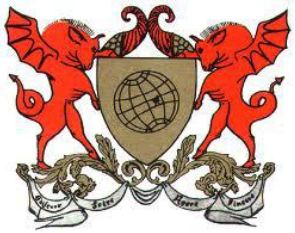
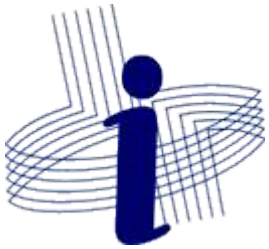


Universidade Federal de Viçosa
Departamento de Informática
Centro de Ciências Exatas e Tecnológicas



INF 112
Programação 2
Arquivos em C++ - Parte 3

Arquivos em C++ - Parte 3

- Em C++ é possível utilizar a API da linguagem C.
- Nesta aula, veremos alguns comandos básicos para manipulação de arquivos utilizando a API C e, então, será apresentada uma técnica para realizar “lock” em arquivos.

Arquivos em C++ - Parte 3

- Em, C, os arquivos normalmente são acessados utilizando o tipo “FILE”.
- Mais especificamente, utiliza-se um “apontador para arquivo” para representar o arquivo e um conjunto de funções para realizar operações no arquivo.
- Funções:
  - Abrir arquivo
  - Fechar arquivo
  - Ler/gravar dados em arquivo

Arquivos em C++ - Parte 3

- Abrir arquivo.
  - Os arquivos são abertos utilizando a função fopen().
  - Ela retorna um apontador (FILE *).
  - Recebe como parâmetro o nome do arquivo (uma string de C) e o modo de abertura (uma string de C).
  - Retorna NULL se houver algum problema na abertura do arquivo.
  - Modos de abertura:
    - “r”: modo de leitura
    - “w”: modo de escrita
    - “a”: saída com concatenação
    - “r+”: modo de leitura/escrita (o arquivo deve existir).
    - “w+”: modo de leitura/escrita (o arquivo pode existir – se existir, seu conteúdo será apagado).
    - ....

```
#include <cstdio>
…
FILE *arquivo = fopen(“teste.txt”,”r”);
```

Arquivos em C++ - Parte 3

- Fechar arquivo.
  - Para fechar um arquivo, utiliza-se a função fclose().
  - Ela recebe como parâmetro um apontador para arquivo.

```
#include <cstdio>
…
FILE *arquivo = fopen("teste.txt","r");
..
fclose(arquivo);
```

Arquivos em C++ - Parte 3

- A função fprintf() é utilizada normalmente para se escrever em arquivos.
- Ela recebe como parâmetros um ponteiro para arquivo, uma string de C especificando o formato dos dados a serem gravados no arquivo e um conjunto de parâmetros contendo valores a serem gravados.

```
#include <cstdio>
…
FILE *arquivo = fopen(“teste.txt”,”w”);
int x = 3;
fprintf(arquivo, “O valor de x eh: %d\n”, x);
fprintf(arquivo, “O valor de x eh: %d, e o valor de x+1 eh: %d \n”, x,x+1);
..
fclose(arquivo);
```

Arquivos em C++ - Parte 3

- A string a ser impressa pelo fprintf pode conter texto e, opcionalmente, especificadores de formato que são substituídos (em ordem) pelos valores que são passados como argumento da função.
- Os especificadores de formato são precedidos por um caractere %. Se o usuário quiser imprimir o caractere %, basta utilizar o especificador “%%”.
- Abaixo, temos a tabela com alguns especificadores de formato:

Especificador	Tipo
d	Número inteiro decimal
u	Número inteiro decimal sem sinal (unsigned)
x	Número inteiro hexadecimal
f	Número de ponto flutuante decimal
s	String de caracteres
c	Caractere
lld	Long long

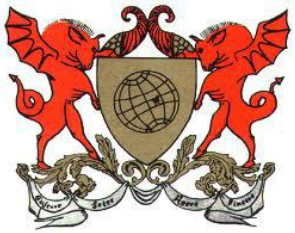
```
#include <cstdio>
int main() {
	FILE *arquivo = fopen("teste.txt","w");
	int x = 12;
	long long y = 20000;
	float w = 3.14159265358979;
	char caractere = '+';
	char str[] = "teste 123";

	fprintf(arquivo, "O valor de x eh: %d\n", x);
	//Note que o primeiro argumento é utilizado no primeiro "%d" e o segundo é utilizado
	//para substituir o segundo "%d"
	fprintf(arquivo, "O valor de x eh: %d, e o valor de x+1 eh: %d \n", x,x+1);
	fprintf(arquivo, "O valor de x em hexa eh: %x\n", x);
	fprintf(arquivo, "O valor de y eh: %lld\n", y);
	fprintf(arquivo, "O valor de w eh: %f \n", w);
	fprintf(arquivo, "O valor do caractere eh: %c \n", caractere);
	fprintf(arquivo, "O valor de str eh: %s \n", str);
	fprintf(arquivo, "Imprimindo %%\n");

	//Pode-se especificar também o número de dígitos impressos (em números reais)
	fprintf(arquivo, "O valor de str eh: %.1f \n", w);
	fprintf(arquivo, "O valor de str eh: %.4f \n", w);
	fprintf(arquivo, "O valor de str eh: %.10f \n", w);

	fclose(arquivo);
	return 0;  }
```

Arquivos em C++ - Parte 3

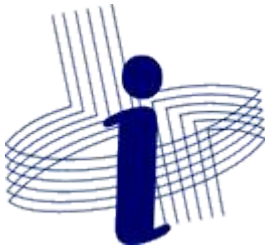
- Para a leitura de dados em arquivos, há várias funções disponíveis na API C.
- Por exemplo a função fscanf() recebe como parâmetros um apontador para arquivos, uma string contendo designadores e apontadores para as variáveis onde armazenar os valores.
- Veja um exemplo de uso no slide seguinte.

```
int main() {
	FILE *arquivo = fopen("testeR.txt","r");
	int n;
	while(true) {
		//Note que, como foi utilizado um designador
		//para int (%d), o terceiro argumento é um
		//apontador para int...
		fscanf(arquivo,"%d", &n);
		//a funcao feof verifica se o fim de arquivofoi atingido...
		if (feof(arquivo)) break;
		//O printf imprime na saída padrão (igual cout)
		printf("Lido: %d\n",n);
	}
	fclose(arquivo);
	return 0;
}
```

```
$ cat testeR.txt
5
10
15
$ ./a.out
Lido: 5
Lido: 10
Lido: 15
```



Universidade Federal de Viçosa
Departamento de Informática
Centro de Ciências Exatas e Tecnológicas



Lock em arquivos

Voltando (temporariamente) ao uso de streams da API C++....

Arquivos em C++ - Parte 3

- Considere o seguinte programa:

```
void gravaRegistro(ofstream &fout, double preco, int quantidade, int codigoItem, int idGravador) {
	fout << "id" << idGravador << " Produto: " << codigoItem << " { " << endl;
	fout << "id" << idGravador << "		Quantidade:" << quantidade << endl;
	fout << "id" << idGravador << "		Preco:" << preco << endl;
	fout << "id" << idGravador << "		Total:" << quantidade*preco << endl;
	fout << "id" << idGravador << " }" << endl;
}

int main(int argc, char **argv) {
	int id=atoi(argv[1]);
	int n = atoi(argv[2]);

	ofstream fout("teste.txt", ios::app);

	srand(time(NULL));
	for(int i =0;i<n;i++) {
		gravaRegistro(fout, (rand()%1000)*1.0/100 , rand()%100, rand()%100, id);
	}
	fout.close();
}
```

Arquivos em C++ - Parte 3

- A ideia do programa é gravar vários registros contendo informações sobre produtos: código, preço e quantidade.
- Por motivos de simplicidade, são utilizados dados aleatórios (note que podem haver vários registros com códigos repetidos).
- Antes de imprimir cada linha do registro, também é impresso um código (id) que será utilizado para identificar a execução do programa.

```
$ ./a.out 1 3
$ ./a.out 5 2
$ cat teste.txt
id1 Produto: 69 {
id1		Quantidade:39
id1		Preco:1.24
id1		Total:48.36
id1 }
id1 Produto: 12 {
id1		Quantidade:93
id1		Preco:3.97
id1		Total:369.21
id1 }
id1 Produto: 33 {
id1		Quantidade:42
id1		Preco:9.95
id1		Total:417.9
id1 }
id5 Produto: 18 {
id5		Quantidade:25
id5		Preco:7.52
id5		Total:188
id5 }
id5 Produto: 8 {
id5		Quantidade:54
id5		Preco:9.27
id5		Total:500.58
id5 }
```

Arquivos em C++ - Parte 3

- Imagine que seja desenvolvido um programa similar ao anterior, mas cujo objetivo seja acessar um determinado site da internet e coletar dados sobre os produtos à venda nesse site (os dados coletados serão gravados em um arquivo).
- Assim, o programa iria ser mantido em execução e o arquivo com os resultados seria “povoado” pelo programa.
- Imagine que o usuário queira coletar simultaneamente dados sobre vários sites distintos (tais dados seriam gravados em um mesmo arquivo).
- O que aconteceria se várias instâncias desse programa fossem executadas simultaneamente (ou seja,o usuário iria manter em execução um processo desse programa para cada site a ser analisado)?

```
$ ./a.out 5 5000 & ./a.out 1 5000
$ cat teste.txt
id5 Produto: 3 {
id1 Produto: 3 {
id5             Quantidade:30
id1             Quantidade:30
id5             Preco:4.07
id1             Preco:4.07
id5             Total:122.1
id1             Total:122.1
id5 }
id1 }
….
id5 Produto: 49 {
id1             Preco:5.54
id5             Quantidade:74
id1             Total:55.4
id5             Preco:1.18
id1 }
id5             Total:87.32
id1 Produto: 19 {
id5 }
id1             Quantidade:48
…….....
```

Arquivos em C++ - Parte 3

- Problemas parecidos ocorrem quando vários processos (programas em execução) podem tentar acessar um mesmo arquivo simultaneamente.
- Uma forma de resolver isso é utilizando algum tipo de “controle de concorrência”.
- Nesta aula, vamos dar uma breve noção de controle de acesso para coordenar acessos a arquivos.
- Ao realizar controle de acessos a um arquivo, um processo consegue “bloquear” (dar “*lock*”) o acesso (leitura/gravação) a um arquivo (ou a parte de um arquivo) no qual esse processo está realizando alguma operação.

Arquivos em C++ - Parte 3

- No Linux, a função *flock* pode ser utilizada para dar lock em arquivos.
- A função *flock* recebe como parâmetro um inteiro que identifica o arquivo (chamado de descritor de arquivo) e um inteiro que define qual operação de *lock* será realizada.
- O identificador do arquivo pode ser obtido utilizando a função *fileno* , que recebe como parâmetro um ponteiro para um arquivo aberto e retorna o descritor do arquivo em questão.

```
#include <sys/file.h>

FILE *arquivo = fopen("teste.txt", "a");
cout << “Identificador do arquivo: “ <<  fileno(arquivo) << endl;
```

Arquivos em C++ - Parte 3

- As seguintes operações (definidas por contantes) estão disponíveis:
  - LOCK_SH: Adiciona ao arquivo um *lock* compartilhado – mais de um processo pode ter locks compartilhados ao mesmo tempo.
  - LOCK_EX: Adiciona ao arquivo um lock exclusivo – só um processo pode ter um lock exclusivo em um determinado momento.
  - LOCK_UN: Remove todos os locks do arquivo (todos os colocados pelo processo corrente).
- Um arquivo não pode ter simultaneamente locks compartilhados e exclusivos.
- Se um processo tentar dar *lock* em um arquivo e esse arquivo já possuir um *lock* não compatível (ex: se o processo tentar dar um *lock* exclusivo em um arquivo que já possui *lock* exclusivo ou *lock* compartilhado), a função *flock* "bloqueia" o processo (ou seja, o processo fica esperando o *lock* do arquivo ser removido).
- A seguir, temos o código de impressão de "registros" implementado com o uso de *locks*.

```
#include <sys/file.h>
using namespace std;
void gravaRegistro(FILE *arquivo, double preco, int quantidade, int codigoItem, int idGravador) {
        fprintf(arquivo, "id%d Produto: %d { \n", idGravador, codigoItem);
        fprintf(arquivo, "id%d         Quantidade : %d \n", idGravador, quantidade);
        fprintf(arquivo, "id%d         Preco: %lf\n", idGravador, preco);
        fprintf(arquivo, "id%d         Total: %lf\n", idGravador, quantidade*preco );
        fprintf(arquivo, "id%d }\n", idGravador);
        fflush(arquivo);
}
int main(int argc, char **argv) {
        int id=atoi(argv[1]);
        int n = atoi(argv[2]);
        FILE *arquivo = fopen("teste.txt", "a");
        srand(time(NULL));
        for(int i =0;i<n;i++) {
                //Da um “lock” exclusivo no arquivo para realizar gravação de dados...
                flock(fileno(arquivo) , LOCK_EX);
                gravaRegistro(arquivo, (rand()%1000)*1.0/100 , rand()%100, rand()%100, id);
                flock(fileno(arquivo) , LOCK_UN); //Remove o lock...

        }
        fclose(arquivo);
        return 0;
} // *** Execute e abra o arquivo para ver a ordem dos Id's
```

Arquivos em C++ - Parte 3

- Os *locks* utilizando *flock* são apenas “aconselhadores”.
- Isso significa que processos PODEM utilizar os locks para gerenciar os acessos a arquivos.
- Porém, se um determinado processo tentar acessar um arquivo (que contém lock) sem verificar o *lock*, ele não será impedido!
- Note que no exemplo anterior utilizamos a função *fflush* para esvaziar o buffer. Isso foi necessário pois os dados enviados ao *fprintf* são *bufferizados.* Assim, pode acontecer do buffer ser esvaziado após o processo sendo executado tiver liberado o *lock* !