Aula de Arquivos

Jesimar da Silva Arantes

Instituto de Ciências Matemáticas e de Computação
Universidade de São Paulo - São Carlos, SP

Estagiário PAE: Jesimar da Silva Arantes
Professor: Claudio Fabiano Motta Toledo

Março – 2016

Arquivos

- São estruturas de dados armazenadas fora da memória principal, geralmente no HD, CD, DVD, pen-drive, etc.
- Os arquivos são usados, principalmente quando o conjunto de informações a serem trabalhadas é numeroso ou quando é necessário o armazenamento permanente.

Vantagens

- Armazenamento durável;
- Permite armazenar uma grande quantidade de informação;
- Acesso concorrente aos dados.

Classificações

- Classificações quanto ao formato:
  - Arquivo Binário
  - Arquivo de Texto
- Classificações quanto a forma de acesso:
  - Arquivo sequêncial
  - Arquivo de acesso aleatório

Arquivos

- Um arquivo deve possuir um nome
- Um arquivo geralmente possui uma extensão (mas, não é necessário)
- Cuidado: a extensão do arquivo não define o seu tipo
- O que define um arquivo é a maneira como os dados estão organizados e as operações usadas por um programa para processar ler ou escrever esse arquivo.

- A linguagem C vê cada arquivo como uma sequência de bytes
- Em C não existem **comandos** de Entrada e Saída
- Estas tarefas são executadas por **funções** especialmente criadas para esta finalidade.
- As principais funções estão armazenadas na biblioteca **stdio.h**
- Dados podem ser manipulados em dois diferentes tipos de fluxos (*streams*):
  - Arquivos de Texto (fluxo de texto);
  - Arquivos Binários (fluxo binário).

Classifique as Extensões em Texto e Binário

- txt
- doc
- mp4
- pdf
- mp3
- csv
- xls
- jpg
- png
- c
- html
- sh
- avi
- tex
- bat
- bin
- xml
- dat
- sgl
- jsa

Resposta das Extensões

- Texto
  - txt
  - csv
  - c
  - html
  - sh
  - bat
  - tex
  - xml
  - dat
  - sgl
  - jsa
- Binário
  - doc
  - mp4
  - pdf
  - mp3
  - xls
  - jpg
  - png
  - avi
  - bin
  - dat
  - jsa

- **Stream de texto:** composto por uma sequência de caracteres, que pode ou não ser dividida em linhas terminadas por um caractere de final de linha.
  - Dados gravados exatamente como seriam impressos na tela;
  - Os dados são gravados como caracteres de 8 bits utilizando a tabela ASCII;
  - Para isso existe uma etapa de conversão (tradução) de dados;
  - Um detalhe que deve ser considerado é que na última linha não é obrigatório o caractere de fim de linha;
  - Cada linha contém zero ou mais caracteres;
  - Uma linha não é uma string, terminação da string '\0';
  - Uma linha é terminada por '\n';
  - Problemas com a conversão:
    - Arquivos em geral maiores;
    - Operações de leituras e escritas mais lentas.

Arquivos

Fluxos ou Streams

Bin	Oct	Dec	Hex	Sinal
0010 0000	040	32	20	(espaço)
0010 0001	041	33	21	!
0010 0010	042	34	22	"
0010 0011	043	35	23	#
0010 0100	044	36	24	$
0010 0101	045	37	25	%
0010 0110	046	38	26	&
0010 0111	047	39	27	'
0010 1000	050	40	28	(
0010 1001	051	41	29	)
0010 1010	052	42	2A	*
0010 1011	053	43	2B	+
0010 1100	054	44	2C	,
0010 1101	055	45	2D	-
0010 1110	056	46	2E	.
0010 1111	057	47	2F	/
0011 0000	060	48	30	0
0011 0001	061	49	31	1

Bin	Oct	Dec	Hex	Sinal
0100 0000	100	64	40	@
0100 0001	101	65	41	A
0100 0010	102	66	42	B
0100 0011	103	67	43	C
0100 0100	104	68	44	D
0100 0101	105	69	45	E
0100 0110	106	70	46	F
0100 0111	107	71	47	G
0100 1000	110	72	48	H
0100 1001	111	73	49	I
0100 1010	112	74	4A	J
0100 1011	113	75	4B	K
0100 1100	114	76	4C	L
0100 1101	115	77	4D	M
0100 1110	116	78	4E	N
0100 1111	117	79	4F	O
0101 0000	120	80	50	P
0101 0001	121	81	51	Q

Bin	Oct	Dec	Hex	Sinal
0110 0000	140	96	60	`
0110 0001	141	97	61	a
0110 0010	142	98	62	b
0110 0011	143	99	63	c
0110 0100	144	100	64	d
0110 0101	145	101	65	e
0110 0110	146	102	66	f
0110 0111	147	103	67	g
0110 1000	150	104	68	h
0110 1001	151	105	69	i
0110 1010	152	106	6A	j
0110 1011	153	107	6B	k
0110 1100	154	108	6C	l
0110 1101	155	109	6D	m
0110 1110	156	110	6E	n
0110 1111	157	111	6F	o
0111 0000	160	112	70	p
0111 0001	161	113	71	q

Figure 1: Tabela ASCII. Fonte: https://pt.wikipedia.org/wiki/ASCII

- ASCII: sistema de codificação de caracteres
  - Codifica um conjunto de 128 sinais
  - 95 sinais gráficos (letras do alfabeto latino, sinais de pontuação e sinais matemáticos)
  - 33 sinais de controle
  - Usa apenas 7 bits
  - No entanto, ocupa um byte (8 bits)
- Unicode: sistema de codificação de caracteres
  - Codifica um conjunto de mais de 107 mil caracteres
  - Cobre quase todos os sistemas de escrita
  - Alfabeto latino, grego, arabe, japonês, chinês, braile, hebraico, élfico
  - Escritas aguardando codificação: hieróglifos egípcios e maias, babilônico
  - Usa 32 bits e ocupando quatro bytes.

- **Stream binário:** composto por uma sequência de bytes lidos, **sem tradução**, diretamente do dispositivo externo.
  - Os dados são gravados exatamente como estão organizados na memória do computador;
  - Não ocorre nenhuma conversão (tradução) dos dados;
  - Normalmente utilizado para salvar dados que serão lidos pelo mesmo programa;
  - Blocos de dados da memória podem ser lidos/gravados diretamente de/para arquivo;
  - Vantagens da não conversão:
    - Arquivos em geral menores;
    - Operações de leituras e escritas mais rápidas.

Arquivos

Exemplo de Tradução

- Gravar um valor inteiro igual a 101 em um arquivo de texto
- Gravar um valor inteiro igual a 101 em um arquivo binário

Arquivos

Exemplo de Tradução

- Gravar um valor inteiro igual a 101 em um arquivo de texto
  - Converte o valor 1 usando a tabela ASCII
  - Converte o valor 0 usando a tabela ASCII
  - Converte o valor 1 usando a tabela ASCII
  - Resultado: ???
- Gravar um valor inteiro igual a 101 em um arquivo binário
  - Conversão de 101 para binário
  - Escrever esse valor no arquivo
  - Resultado: ???
  - OBS: essa conversão não existe explicitamente, pois o valor vem da memória já em binário

Arquivos

Nome de arquivos

- Os nomes são armazenados em strings
- Deve-se seguir as regras de nomenclatura do sistema operacional
  - Note que no Linux o caminho é indicado pela barra normal
  - Note que no Windows o caminho é indicado pela barra invertida
- O nome do arquivo pode ser informado junto com sua localização no disco
  - Caminho relativo: Ex: dados.txt
  - Caminho absoluto (Linux): Ex: /home/jesimar/programa/dados.txt
  - Caminho absoluto (Windows): Ex: c:\programa\dados.txt
- Como em C a barra invertida também tem um significado especial, para representar o caminho numa string é necessário preceder cada barra com outra barra invertida
  - Linux: char *filename = "/home/jesimar/programa/dados.txt";
  - Windows: char *filename = "c:\\programa\\dados.txt";

- Operações comuns em arquivos são:
  - abertura e fechamento de arquivos;
  - apagar um arquivo;
  - leitura e escrita de um caractere;
  - indicação de que o fim do arquivo foi atingido;
  - posicionar o arquivo em um ponto determinado.
- Importante:
  - Ao final das operações necessárias o programa deve fechar o arquivo;
  - Quando um programa é encerrado todos os arquivos associados são fechados automaticamente e os conteúdos dos buffers são descarregados para o dispositivo externo.

FILE *arq;
Declara um ponteiro para arquivo (FILE)

arq = fopen("nome_do_arq", modo);
Abre/cria arquivo. Retorna NULL se ocorrer algum erro.

fclose(arq);
Fecha um arquivo

Arquivos

Modo de Abertura/Criação de Arquivo

Modo	Significado
"r"	Abre um arquivo texto para leitura. O arquivo deve existir antes de ser aberto.
"w"	Abrir um arquivo texto para gravação. Se o arquivo não existir, ele será criado. Se já existir, o conteúdo anterior será destruído.
"a"	Abrir um arquivo texto para gravação. Os dados serão adicionados no fim do arquivo ("append"), se ele já existir, ou um novo arquivo será criado, no caso de arquivo não existente anteriormente.
"rb"	Abre um arquivo binário para leitura. Igual ao modo "r" anterior, só que o arquivo é binário.
"wb"	Cria um arquivo binário para escrita, como no modo "w" anterior, só que o arquivo é binário.
"ab"	Acrescenta dados binários no fim do arquivo, como no modo "a" anterior, só que o arquivo é binário.
"r+"	Abre um arquivo texto para leitura e gravação. O arquivo deve existir e pode ser modificado.
"w+"	Cria um arquivo texto para leitura e gravação. Se o arquivo existir, o conteúdo anterior será destruído. Se não existir, será criado.
"a+"	Abre um arquivo texto para gravação e leitura. Os dados serão adicionados no fim do arquivo se ele já existir, ou um novo arquivo será criado, no caso de arquivo não existente anteriormente.
"r+b"	Abre um arquivo binário para leitura e escrita. O mesmo que "r+" acima, só que o arquivo é binário.
"w+b"	Cria um arquivo binário para leitura e escrita. O mesmo que "w+" acima, só que o arquivo é binário.
"a+b"	Acrescenta dados ou cria uma arquivo binário para leitura e escrita. O mesmo que "a+" acima, só que o arquivo é binário

Arquivos

Funções da biblioteca stdio.h

Função	Descrição
fopen()	Abre um arquivo
fclose()	Fecha um arquivo
fseek()	Posiciona o ponteiro do arquivo
feof()	Retorna VERDADE se chegou ao fim do arquivo
fflush()	Descarrega o buffer associado com o arquivo
fprintf()	Equivalente a printf(), usando stream
fscanf()	Equivalente a scanf(), usando stream
fgets()	Equivalente a gets(), usando stream

- fprintf(arq, “string”, variáveis)
  escreve no arquivo de texto (semelhante ao printf).
- fscanf(arq, “string”, variáveis)
  lê do arquivo de texto (semelhante ao scanf).
- fgets(*dados, tamanho dos dados, arq)
  lê do arquivo de texto (semelhante ao gets).
- fwrite(*dados, tamanho dos dados, número de itens, arq)
  escreve dados no arquivo binário.
- fread(*dados, tamanho dos dados, número de itens, arq)
  lê dados do arquivo binário.

- Escrever um programa em C que escreve os números inteiros de 1 até 100 em um arquivo de texto, pulando uma linha entre cada número lido.

Arquivos

Exemplo Escrever Arquivo de Texto

```c
#include <stdio.h>

int main(){
    char name[] = "dados.txt";
    FILE *arq = fopen(name, "w");
    if (arq != NULL){
        for (int i = 0; i < 100; i++){
            fprintf(arq, "i = %d\n", i);
        }
    } else{
        printf("erro\n");
        return -1;
    }
    fclose(arq);

    return 0;
}
```

Figure 2: Exemplo de código que escreve em arquivo de texto.

Saída: ???

- Escrever um programa em C que leia os números inteiros de 1 até 100, gravados no arquivo de texto e os imprima na tela.

Arquivos

Exemplo Ler Arquivo de Texto

```c
#include <stdio.h>

int main(){
    char name[] = "ler.txt";
    FILE *arq = fopen(name, "r");
    if (arq != NULL){
        while (!feof(arq)){
            int value;
            fscanf(arq, "%d", &value);
            printf("%d\n", value);
        }
    } else{
        printf("erro\n");
        return -1;
    }
    fclose(arq);
    return 0;
}
```

Figure 3: Exemplo de código que lê em arquivo.

- Escrever um programa em C que escreve os números inteiros de 1 até 100, em um arquivo binário.

Arquivos

Exemplo Escrever Arquivo Binário

```c
#include <stdio.h>

int main(){
    char name[] = "dados.bin";
    FILE *arq = fopen(name, "wb");
    if (arq != NULL){
        for (int i = 0; i < 100; i++){
            fwrite(&i, sizeof(int), 1, arq);
        }
    } else{
        printf("erro\n");
        return -1;
    }
    fclose(arq);
    return 0;
}
```

- Escrever um programa em C que leia os números inteiros de 1 até 100, gravados no arquivo binário anterior.

Arquivos

Exemplo Ler Arquivo Binário

```c
#include <stdio.h>

int main(){
    char name[] = "dados.bin";
    FILE *arq = fopen(name, "rb");
    if (arq != NULL){
        while(!feof(arq)){
            int i;
            fread(&i, sizeof(int), 1, arq);
            printf("%d ", i);
        }
    } else{
        printf("erro\n");
        return -1;
    }
    fclose(arq);
    return 0;
}
```

- Escrever um programa em C que armazena uma struct de dados de um aluno contendo (nome[40], idade, sexo, e matrícula), em um arquivo binário.

- Escrever um programa em C que leia a struct de dados aluno contendo os campos (nome[40], idade, sexo, e matrícula), do arquivo binário e imprima na tela as informações.
- Escrever um programa em C que leia a struct de dados aluno contendo os campos (nome[40], matrícula, sexo, e idade), do arquivo binário e imprima na tela as informações.

Arquivos

Funções para Manipulação de Arquivo

- fputc(char, arq)
  escreve no arquivo (semelhante ao putc).
- fgetc(arq)
  lê do arquivo (semelhante ao getc).
- fputs(string, arq)
  escreve no arquivo (semelhante ao puts).
- fgets(string, tam, arq)
  lê do arquivo (semelhante ao gets).

- feof(arq)
  retorna zero se ponteiro não aponta para o fim do arquivo.
- fflush(arq)
  descarrega o buffer no arquivo.
- fseek(arq, posição a ser buscada, posição de referencia)
  busca posição em arquivo aberto "posição a ser buscada": em bytes
  posição de referência:
  - SEEK_SET para início do arquivo;
  - SEEK_CUR para posição atual;
  - SEEK_END para fim do arquivo.

```c
#include <stdio.h>

int main(){
    FILE * arq = fopen( "exemploFseek.txt" , "w" );
    if (arq != NULL){
        fputs("Testando funcao fseek" , arq);
        fseek(arq , 12 , SEEK_SET);
        fputs("cionalidade" , arq);
    }else {
        printf("erro\n");
        return -1;
    }
    fclose(arq);
    return 0;
}
```

Figure 4: Exemplo de código que usa função fseek

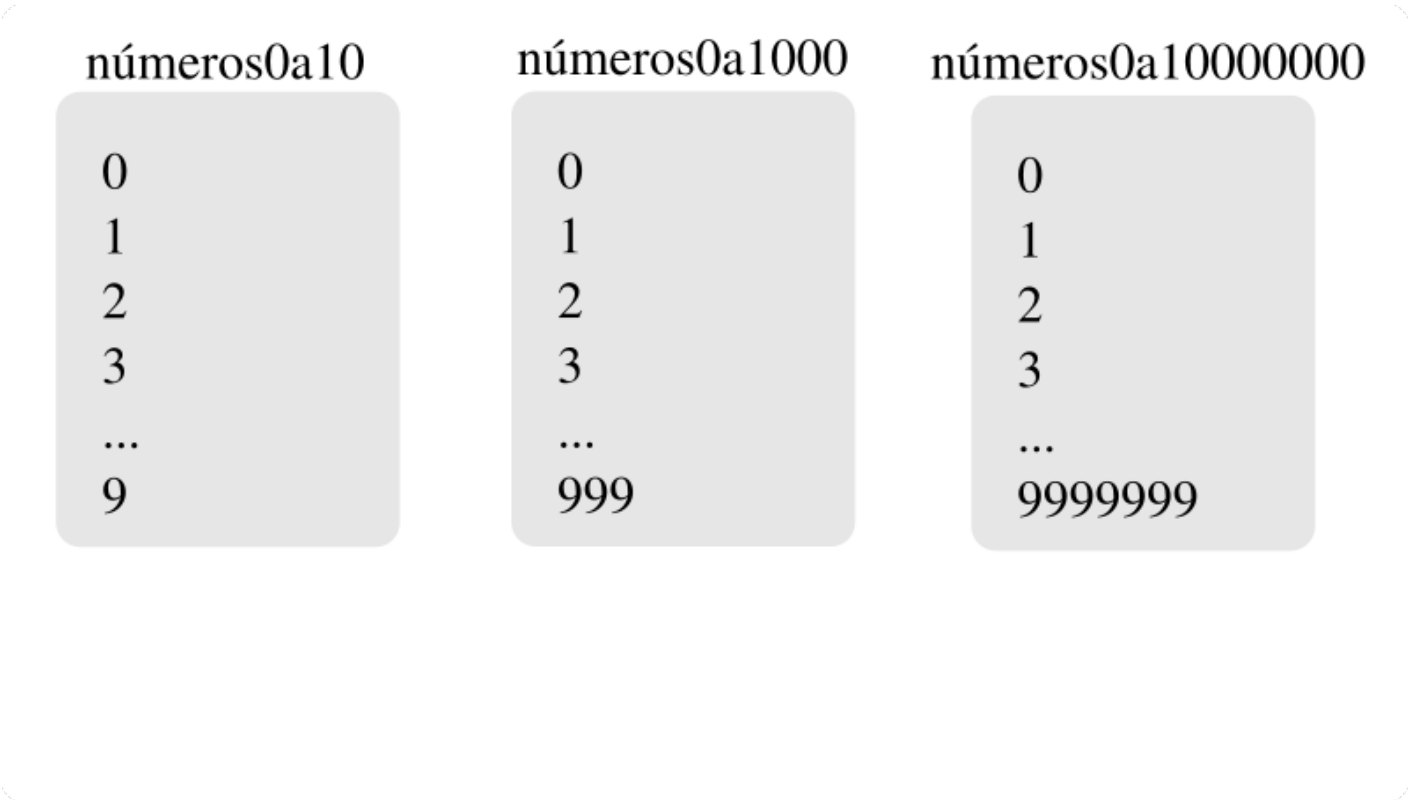
Saída: ???

- rewind(arq)
  posiciona o cursor de volta ao início do arquivo.
- remove(nome_arquivo)
  apaga um arquivo do HD.
- rename(nome_atual, novo_nome)
  troca o nome de um arquivo.

- Escrever um programa em C que leia um arquivo de texto e escreva o conteúdo desse arquivo duas vezes na tela sem fechar o arquivo. Faça um exemplo com o comando rewind(...) e outro exemplo com o comando fseek(...).

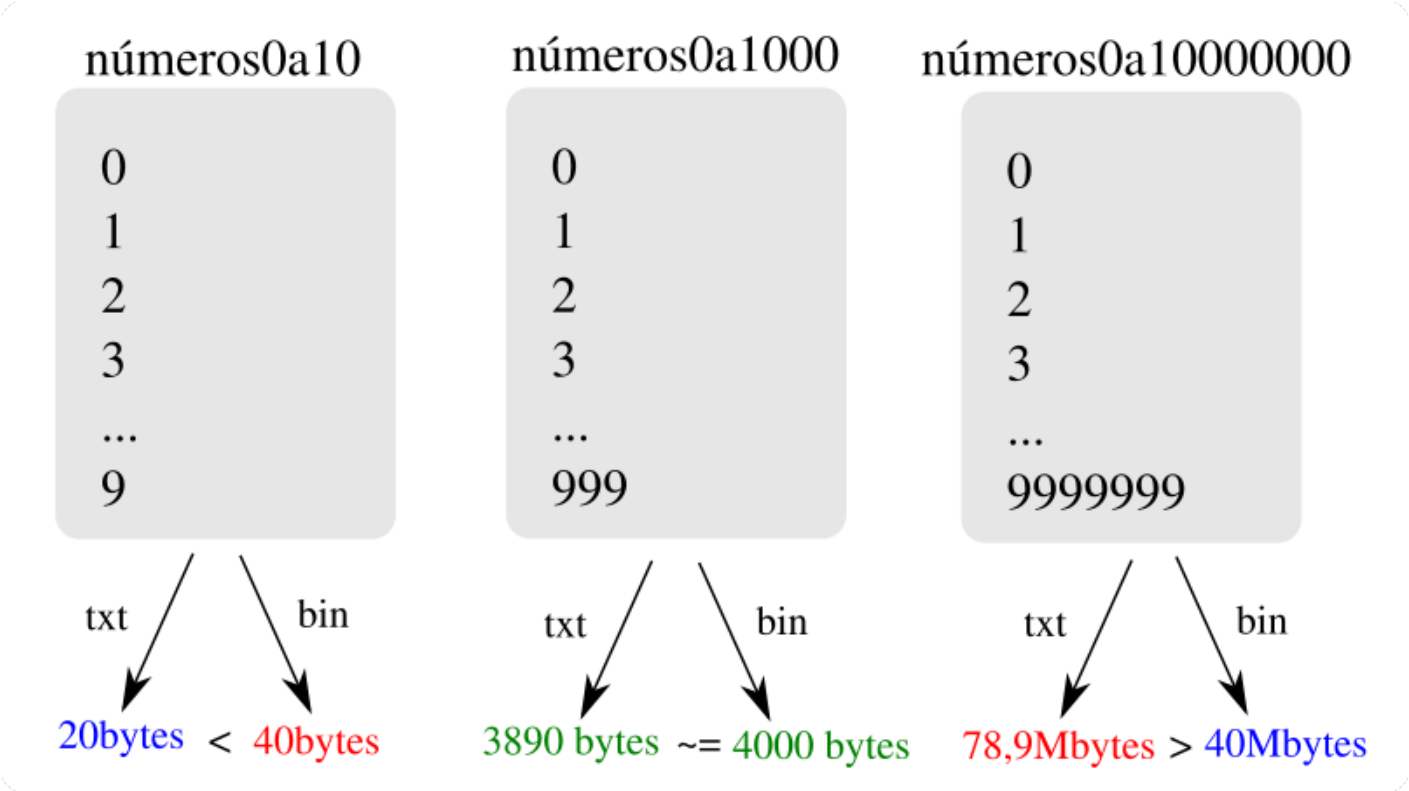
Análises do Tamanho dos Arquivos

- Fazer um programa em C para armazenar um arquivo binário e um arquivo texto capaz de armazenar valores inteiros como apresentado no esquema abaixo. Em seguida, analize os tamanhos dos arquivos.



Análises do Tamanho dos Arquivos

- Fazer um programa em C para armazenar um arquivo binário e um arquivo texto capaz de armazenar valores inteiros como apresentado no esquema abaixo. Em seguida, analize os tamanhos dos arquivos.



- Em geral os arquivos txt são maiores, mas depende muito da aplicação e da implementação.

Sugestões

- Instalar algum editor hexadecimal
  - Windows: HxD
  - Linux: Okteta
- Abrir diferentes arquivos binários e de texto com esse editor e analizá-los

Referências

- Notas de aula professor Denis Wolf.
- Notas de aula professor Jó Ueyama.
- http://www.cplusplus.com/reference/clibrary/