UNIDADE 2 – ARQUIVOS DE TEXTO, DOCUMENTOS, PLANILHAS E APRESENTAÇÕES

MÓDULO 1 – ARQUIVOS DE TEXTO – PARTE 1

01

1 - DIFERENCIANDO ARQUIVOS DE TEXTO DE DOCUMENTOS TEXTUAIS

Olá, seja bem-vindo a este primeiro módulo da unidade II. Na unidade passada vimos diversos conceitos sobre o SIGAG e sobre o e-Arq. Vimos informações de alto nível sobre como um sistema deve gerenciar documentos para usuários em uma organização.

A partir deste módulo, iremos nos aprofundar nas características técnicas dos arquivos e documentos. Veremos como eles são criados e manipulados pelos programas. Você irá aprender também como programas de computador tratam esses arquivos. Futuramente, quando você aprender a programar em uma determinada linguagem de programação, os conhecimentos adquiridos nesta e nas próximas unidades formarão a base conceitual que você precisará saber para poder manipular arquivos e documentos.

Atenção: para você mesmo fazer testes e verificar os exemplos citados neste módulo, utilizaremos um software livre para edição de arquivos. Como sugestão, baixe e instale o Notepadd++, disponível gratuitamente neste endereço: http://notepad-plus-plus.org/.

02

Antes de entrar nos conceitos dessa unidade você precisa saber muito bem a diferença entre arquivos de texto (texto plano) e de documentostextuais (texto rico).

Documentos textuais são aqueles que você escreve utilizando programas do tipo “editores de texto”, como o Microsoft Word ou o Write da Open Office.

Quando você cria documentos textuais, você pode escolher o tamanho e o tipo da fonte, se é negrito, sublinhado, cores, mudar a formatação dos parágrafos etc. Documentos textuais têm como objetivo criar um documento semelhante aos documentos em papel, como uma carta, um contrato, um livro, uma redação, e outros. Geralmente eles têm a mesma disposição gráfica de uma folha de papel, e por isso, quando são impressos, eles são exatamente iguais ao que vemos na tela. O arquivo que corresponde a um documento textual, além de conter o próprio texto em si, tem também diversas informações estruturais e de formatação, e por isso só podem ser abertos nas ferramentas próprias de edição de documentos textuais.

Já os **arquivos de texto** são documentos que só contêm caracteres (letras, números e símbolos ASCII), arquivos de texto não tem formatação alguma, nem de fonte, nem número de página. Devido à sua simplicidade arquivos de texto são comumente utilizados para armazenamento de informações.

Eles evitam alguns dos problemas encontrados com outros formatos de arquivo, tais como:

- incompatibilidade de abertura,
- interpretação de caracteres acentuados,
- interpretação de caracteres especiais de diversos idiomas.

Além disso, quando a corrupção de dados ocorre em um arquivo de texto, é muitas vezes mais fácil de recuperar e continuar o tratamento do restante do conteúdo.

03

Os arquivos de texto podem ser livremente intercambiados entre os diversos sistemas operacionais, como o Unix, Machintosh, Microsoft Windows, Ms DOS e outros. Alguns detalhes podem necessitar de ajustes como os marcadores de final de linha e de final de arquivo (falaremos mais sobre isso adiante).

Os documentos de texto têm uma infinidade de aplicações:

- Criar textos simples que não precisam de formatação, com arquivos .TXT.
- Criar arquivos de configuração, como os arquivos .INI, .CNF, .CONF ou .CFG.
- Criar código fonte de programas de computador, como os arquivos .PHP, .C, .CPP, .BAS, .VB, .ASP, .Java etc.
- Criar arquivos de log, como os arquivos .LOG.
- Criar arquivos de dados, como os arquivos XML, que podem conter dados e/ou configurações.
- Criar scripts como os arquivos .VBS, .BAT, .CMD, .JS.
- Criar arquivos de exportação e troca de dados como os arquivo .CSV.
- Criar páginas web, como os arquivos .HTM e .HTML.

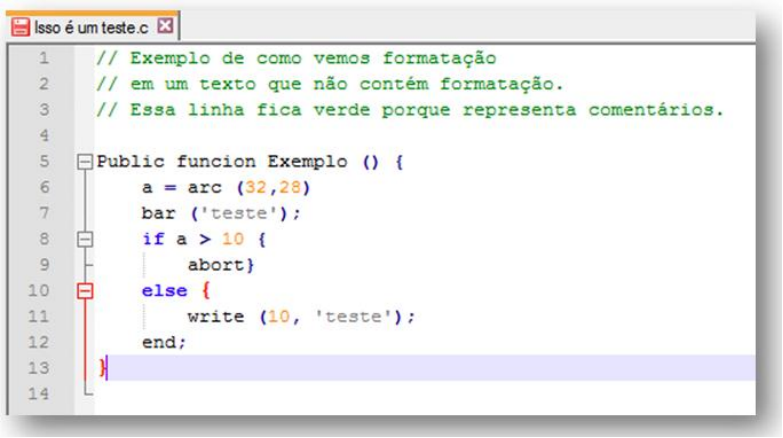
O site da Wikipédia registra mais de 65 tipos de arquivos de diferentes usos para arquivos de texto.

Wikipedia
Veja mais aqui: http://en.wikipedia.org/wiki/List_of_file_formats#Document.

04

Todas as linguagens de programação utilizam arquivos de texto puro como base para os códigos fonte dos programas, os eventuais formatos ou cor da fonte que você vê são manipulações estéticas que os aplicativos de programação oferecem.

Exemplo: enquanto o arquivo original contém apenas texto puro sem formatação, quando você utiliza um software de programação, você vê algo parecido com isso aqui:

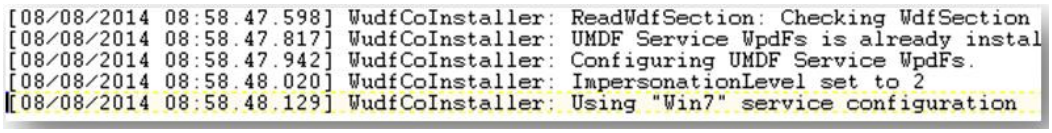


Observe que as formatações em verde, azul, preto ou vermelho não existem e não são controlados pelo usuário, mas sim pelo próprio software de programação.

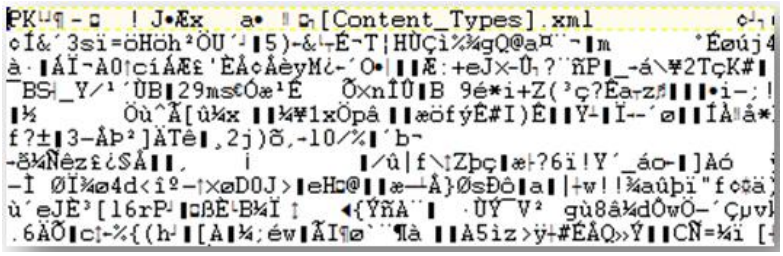
05

1.1 - Arquivo binário e arquivo de texto puro (plano)

Num computador há basicamente dois tipos de arquivos: arquivos binários e arquivos planos. Arquivos planos só possuem letras, números e os símbolos presentes no seu teclado (como, por exemplo: !@#$%^&*_+-). Já os arquivos binários geralmente estão escritos em linguagem de máquina e uma pessoa comum não consegue entender seu conteúdo sem um programa auxiliar que traduza para algo legível. Veja os dois exemplos a seguir:



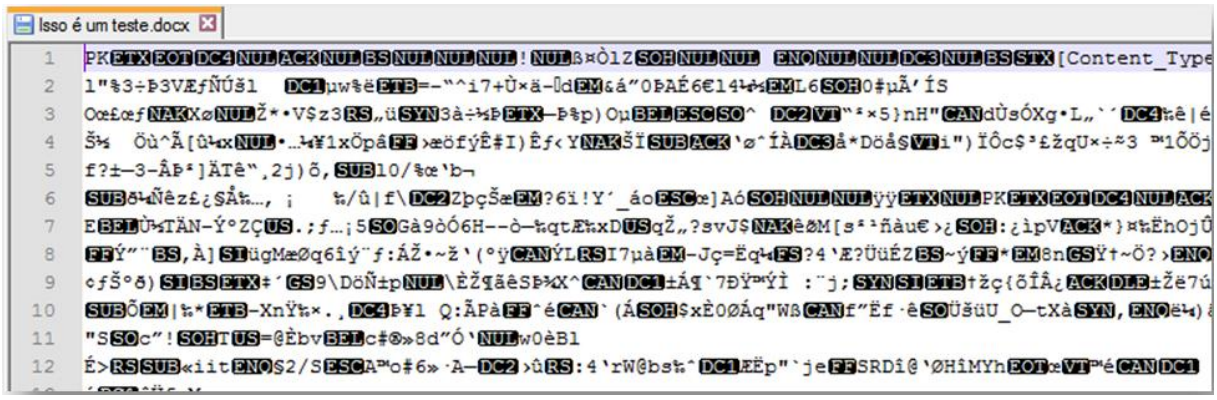
Um arquivo de texto plano é possível um humano ler seu conteúdo sem precisar de um programa que o interprete.



Um arquivo binário é impossível entender seu conteúdo sem um programa que o traduza para a linguagem humana. Esse exemplo acima mostra o conteúdo binário de um documento do Ms Word (.docx).

06

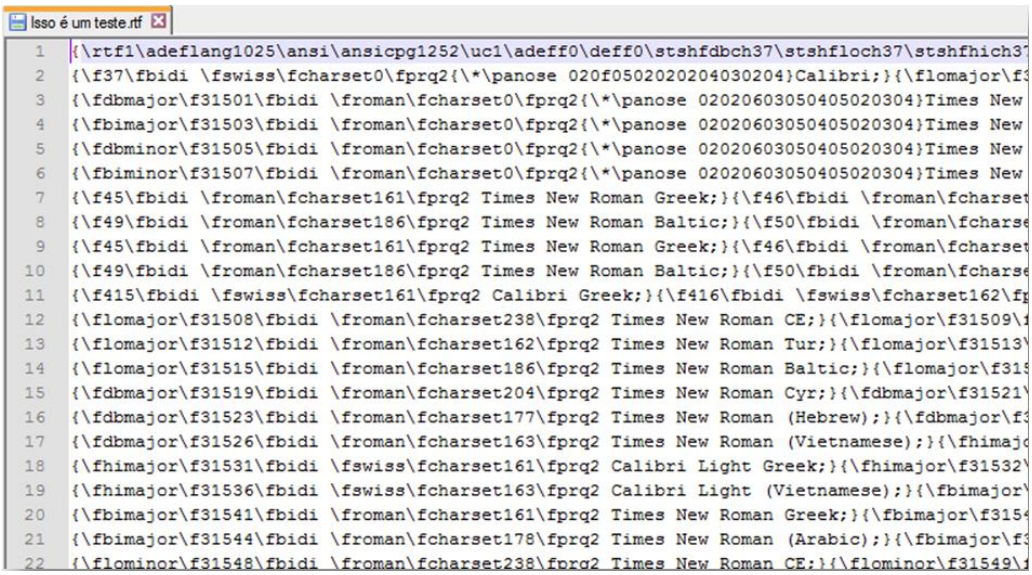
Vamos fazer o outro teste agora. A partir do Ms Word, criaremos um documento com o seguinte conteúdo: “Isso é um teste” e salvaremos esse conteúdo nos seguintes formatos: DocX - Ms Word 2010/13, Doc - Ms Word 2007, RTF e TXT. Vamos comparar o conteúdo desses quatro formatos:



No formato DOCX, o arquivo possui conteúdo binário. Somente com o programa Ms Word conseguiremos entender o que está escrito no arquivo.

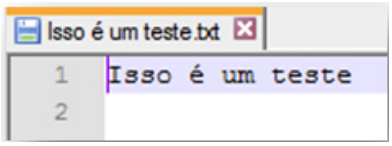
07

Da mesma forma, no formato DOC, o arquivo também possui conteúdo binário. Somente com o programa Ms Word conseguiremos entender o que está escrito no arquivo.


Isso é um teste.rtf

```
{\rtf1\adeflang1025\ansi\ansicpg1252\uc1\adeff0\deff0\stshfdbch37\stshfloch37\stshfhich37
{\f37\fbidi \fswiss\fcharset0\fprq2{\*\panose 020f0502020204030204}Calibri;}{\flomajor\f
{\fdbmajor\f31501\fbidi \froman\fcharset0\fprq2{\*\panose 02020603050405020304}Times New
{\fbimajor\f31503\fbidi \froman\fcharset0\fprq2{\*\panose 02020603050405020304}Times New
{\fdbminor\f31505\fbidi \froman\fcharset0\fprq2{\*\panose 02020603050405020304}Times New
{\fbiminor\f31507\fbidi \froman\fcharset0\fprq2{\*\panose 02020603050405020304}Times New
{\f45\fbidi \froman\fcharset161\fprq2 Times New Roman Greek;}{\f46\fbidi \froman\fcharset
{\f49\fbidi \froman\fcharset186\fprq2 Times New Roman Baltic;}{\f50\fbidi \froman\fcharse
{\f45\fbidi \froman\fcharset161\fprq2 Times New Roman Greek;}{\f46\fbidi \froman\fcharset
{\f49\fbidi \froman\fcharset186\fprq2 Times New Roman Baltic;}{\f50\fbidi \froman\fcharse
{\f415\fbidi \fswiss\fcharset161\fprq2 Calibri Greek;}{\f416\fbidi \fswiss\fcharset162\fp
{\flomajor\f31508\fbidi \froman\fcharset238\fprq2 Times New Roman CE;}{\flomajor\f31509\
{\flomajor\f31512\fbidi \froman\fcharset162\fprq2 Times New Roman Tur;}{\flomajor\f31513
{\flomajor\f31515\fbidi \froman\fcharset186\fprq2 Times New Roman Baltic;}{\flomajor\f31
{\fdbmajor\f31519\fbidi \froman\fcharset204\fprq2 Times New Roman Cyr;}{\fdbmajor\f31521
{\fdbmajor\f31523\fbidi \froman\fcharset177\fprq2 Times New Roman (Hebrew);}{\fdbmajor\f
{\fdbmajor\f31526\fbidi \froman\fcharset163\fprq2 Times New Roman (Vietnamese);}{\fhimaj
{\fhimajor\f31531\fbidi \fswiss\fcharset161\fprq2 Calibri Light Greek;}{\fhimajor\f31532
{\fhimajor\f31536\fbidi \fswiss\fcharset163\fprq2 Calibri Light (Vietnamese);}{\fbimajor
{\fbimajor\f31541\fbidi \froman\fcharset161\fprq2 Times New Roman Greek;}{\fbimajor\f3154
{\fbimajor\f31544\fbidi \froman\fcharset178\fprq2 Times New Roman (Arabic);}{\fbimajor\f
{\flominor\f31548\fbidi \froman\fcharset238\fprq2 Times New Roman CE;}{\flominor\f31549\
```

Já no formato RTF, o arquivo possui conteúdo texto plano. Conseguimos ler algumas informações sem precisar de um programa especial, veja que no fragmento de texto da imagem acima temos informações de fontes de texto utilizadas. Esse formato não está em linguagem de máquina.



Observe que o arquivo no formato texto puro (TXT) contém apenas a informação que queremos e nada mais. Não temos nada que indique uso de fonte, tamanho ou posição. Veja também que o tamanho do arquivo é exatamente o número de caracteres que ele contém, incluindo os espaços e as duas marcações de nova linha (explicaremos essas marcações depois). Como esse arquivo não contém outras informações, ele é o que possui o menor tamanho. Entretanto, formatos como o DOCX podem compactar o conteúdo, o que na prática, para textos muito grandes, o arquivo ficaria menor no formato DOCX do que no formato TXT.

08

2 - CARACTERÍSTICAS GERAIS DE ARQUIVOS DE TEXTO

Todos os arquivos texto são considerados arquivos sequenciais, ou seja, para o computador, o conteúdo do arquivo é uma sequência de caracteres que começa no primeiro e termina no último caractere. O computador não entende que quebras de linhas são necessárias para que o conteúdo seja mais bem apresentado no seu monitor e assim mais fácil de ler. Veja abaixo como nós observamos o conteúdo de um arquivo e como o computador interpreta o mesmo conteúdo.

```
Olá
Esse é um teste
para vermos as quebras de página
ok?
Fim.
```

Texto no formato humano de leitura.

Olá[CR][LF] Esse é um teste[CR][LF] para vermos as quebras de página[CR][LF] ok?[CR][LF] Fim.[CR][LF]

Texto no formato que o computador interpreta.

O que para nós é um texto com várias linhas, para o computador todos os caracteres ocupam uma linha só, por isso o nome **arquivo sequencial**. O que para nós é apenas o final da linha, para o computador são caracteres especiais que marcam uma determinada funcionalidade: a quebra de linha e um novo parágrafo.

09

Então, como fazer para que o texto seja quebrado em linhas?

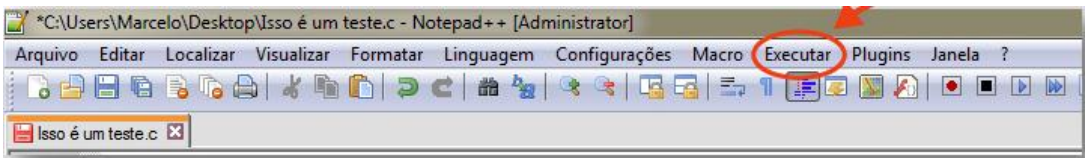
Para documentos no formato do sistema operacional Windows e DOS, há dois caracteres especiais, chamados de "carriage return" (CR) e "Line feed" (LF), representados na tabela ASCII pelos códigos 13 e 10, que indicam que ali é o fim da frase e uma quebra de linha deve ser feita no monitor do usuário. Nós inserimos automaticamente esses caracteres especiais quando pressionamos a tecla "Enter" do teclado.

```
1  Olá[CR][LF]
2  Esse é um teste[CR][LF]
3  para vermos as quebras de página[CR][LF]
4  ok?[CR][LF]
5  Fim.[CR][LF]
```

Veja na tela acima onde estão localizados os caracteres especiais ao final da de cada frase. Esses caracteres determinam que uma nova linha deva ser criada.

Para você visualizar as marcas de final de linha, abra o programa Notepad++, escreva um texto como o exemplo acima (pressionando o "Enter" ao final de cada linha), depois clique no botão de marcação de

caracteres que está na barra de botões do programa: , ou acesse o menu do programa → view → show symbol → show all characters.



Os sistemas operacionais da família Unix (unix, Linux etc.) só utilizam o caractere "line feed" - LF (ASCII 10) para marcar o final da linha, eles não precisam do caractere "carriage return".

10

3 - COMO UM PROGRAMA DE COMPUTADOR MANIPULA UM DOCUMENTO DE TEXTO

É muito comum que um programa de computador precise utilizar arquivos de texto puro para funções auxiliares conforme já citamos. Para manipular os arquivos, os programas de computador normalmente precisam realizar as seguintes operações em um arquivo de texto puro:

- Ler, linha a linha o conteúdo completo do arquivo;
- Ler o conteúdo completo do arquivo de pedaços em pedaços de tamanho pré-determinado (exemplo, ler 1000 caracteres de cada vez);
- Adicionar texto ao final de um arquivo que já existe;
- Criar um arquivo de texto novo, gravando linha a linha seu conteúdo;
- Criar um arquivo de texto novo, gravando pedaços em pedaços de tamanho pré-determinado.
- Renomear arquivos.
- Copiar arquivos para outros lugares.
- Excluir arquivos.

Os métodos que leem/gravam tamanhos fixos são geralmente utilizamos para tratar arquivos binários, mas podem também ser utilizados para tratar arquivos de texto puro.

11

3.1 - Como um programa lê um arquivo de texto.

Observação inicial importante: não se preocupe se você achar a parte de programação que será apresentada neste e no próximo módulo um tanto complicada. Ela usa alguns conceitos que você já aprendeu em outras matérias e outros que você só aprenderá bem detalhadamente quando estudar as matérias de linguagem de programação.O objetivo é apenas fazer uma introdução ao assunto para você entender a lógica de programação (e não a linguagem de programação em si).

Os exemplos que apresentaremos aqui, dependendo da linguagem de programação que você utilizar, poderá possuir algumas variações na estrutura, formato ou nome de métodos. Entretanto, os conceitos de lógica de programação serão sempre os mesmos. Foque seu estudo na lógica apresentada!

Para um programa de computador conseguir ler um arquivo texto, independente da linguagem de programação que você utilizar, provavelmente seu programa vai ter a seguinte **estrutura e conteúdo**:

1. Você precisará de um objeto do tipo “arquivo” (archive) para receber conteúdo arquivo, vamos chamar esse objeto de “oArquivo” e executar o método de abertura de arquivo (open).

```
1  dim oArquivo as Archive;
2  oArquivo.open ("c:\temp\arquivo_de_teste.txt");
```

2. Uma vez aberto, precisaremos criar um loop que irá ler o conteúdo linha a linha, e armazená-lo em uma variável do tipo string (que chamaremos de “sConteudoArquivo”). Para ler o conteúdo, usaremos o método “readline” que lê uma linha de cada vez (ele encontra o CR LF e para nesse ponto). O loop será encerrado quando o arquivo chegar ao final do arquivo. O final

do arquivo é definido como “end of file”, o atributo EOF do objeto arquivo se tornará verdadeiro quando chegar ao final do arquivo. Veja como fica o fragmento do programa agora:

```
4  while not oArquivo.EOF {
5      sConteudoArquivo += oArquivo.ReadLine;}
```

12

3. Ao final, precisaremos fechar o arquivo, chamando o método “close”.

```
7  oArquivo.Close;
```

4. Pronto, o arquivo está fechado e todo o conteúdo dele armazenado na variável **sConteudoArquivo**. Seu programa agora pode fazer outras tarefas que você desejar com o conteúdo do arquivo.

Dessa forma, nosso pequeno programa que lê um arquivo e coloca seu conteúdo dentro de uma variável de texto é o seguinte:

```
1  dim oArquivo as Archive;
2  oArquivo.open ("c:\temp\arquivo_de_teste.txt");
3
4  while not oArquivo.EOF {
5      sConteudoArquivo += oArquivo.ReadLine;}
6
7  oArquivo.Close;
```

13

3.2 - Como um programa cria um arquivo de texto, inserindo informações dentro dele.

Para esse exemplo, vamos supor que nós temos uma variável vetor string, ou seja, em cada dimensão do vetor há uma linha completa. Usaremos o método “inputline” que grava o conteúdo de uma string dentro de um arquivo e ao final adiciona automaticamente os caracteres “CR” e “LF” para marcar a quebra de linha.

Vamos ver agora o código completo e o comentário de cada linha:

```
1  dim sTexto(4) as string
2  sTexto(0) = "olá"
3  sTexto(1) = "Esse é um teste"
4  sTexto(2) = "para vermos as quebras de página"
5  sTexto(3) = "ok?"
6  sTexto(4) = "Fim."
7
8  dim i as integer
9  oArquivo.createfile ("c:\temp\arquivo_de_teste2.txt", text)
10
11 for i = 0 to 4 {
12     oArquivo.inputline (sTexto(i))}
13
14 oArquivo.close
```

- **Na linha 1**, declaramos um vetor de 5 posições (de 0 a 4);
- **Nas linhas 2 a 6**, atribuímos um valor para cada posição do vetor;
- **Na linha 8**, criamos uma variável para contar cada posição do vetor, que usaremos no laço "for" a seguir;
- **Na linha 9**, criamos um arquivo do tipo texto;
- **Na linha 11**, iniciamos o laço "for" que vai repetir e percorrer todas as posições do vetor sTexto.
- **Na linha 12**, gravamos a linha dentro do arquivo, referindo a posição exata do vetor sTexto.
- **Na linha 14**, fechamos o arquivo.

14

3.3 - Como um programa adiciona arquivo de texto, inserindo informações ao final dele.

A ideia é muito semelhante à do item anterior, a única linha que muda é a linha 9, onde ao invés de utilizar um método que cria um arquivo, utilizaremos um método que abre, para gravação, um arquivo que já existe: "openfileforinput". O fragmento do programa ficaria com essa estrutura a seguir:

```
oArquivo.openfileforinput ("c:\temp\arquivo_de_teste2.txt", text)
```

3.4 - Como um programa pode excluir parte do conteúdo de um arquivo.

Comumente, as linguagens de programação não fornecem funcionalidades para excluir parte do conteúdo de um arquivo. Dessa forma, não existem funções básicas que possam permitir:

a) Excluir parte do arquivo.
b) Excluir parte do conteúdo no início do arquivo.
c) Excluir parte do conteúdo no final do arquivo.
d) Excluir parte do conteúdo no meio do arquivo.

A única funcionalidade de exclusão que existe é excluir todo o arquivo. Dessa forma, para realizar as operações citadas na lista acima, o programador deve:

1. Ler todo o conteúdo do arquivo na memória do computador.
2. Excluir o arquivo fisicamente.
3. Criar um novo arquivo, excluindo as informações indesejadas.

15

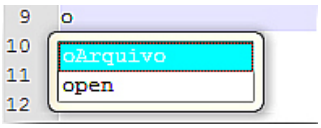
4 - ARQUIVOS FONTE DE PROGRAMAÇÃO

Conforme já falamos, os arquivos fonte de programas de computador são quase todos arquivos texto puro. Entretanto, utilizar um editor de texto genérico pode ser uma péssima ideia para se criar um programa.

Todas as linguagens de programação moderna oferecem programas de edição, chamados interfaces de usuários ou IDE. Há muitas vantagens em se utilizar IDEs próprias para a linguagem alvo que o seu programa está sendo feito. Citamos algumas:

- Validação sintática de código e comandos enquanto você digita, para evitar erros simples de digitação.
- Formatação especial em cores, que facilita a leitura e análise do programa (veja o item 3 deste módulo),
- Funcionalidade de autocompletar que facilita e acelera a digitação.
- Funcionalidades de localizar inteligente, para facilitar a análise e pesquisa de métodos e funções.
- Funcionalidades de compilação, publicação, documentação e depuração integradas (você aprenderá sobre isso na matéria de programação de sistemas.

Exemplo de autocomplete:



Na linha 9, ao digita o caractere “o”, o autocomplete já sugere duas opções: oArquivo e open.

16

5 - Arquivos de log

Arquivos de log registram operações de usuário ou de sistema. Servem para analisar se determinado procedimento realizou todos os passos corretamente ou para auditar atividades de usuários.

Veja alguns exemplos práticos.

Exemplo 1 → Suponha que você esteja desenvolvendo um sistema que realiza uma operação bancária de 10 passos, seria interessante registrar sempre quem fez as operações, em que momento, qual operação foi feita e o resultado da operação. Isso indica que é possível auditar a segurança para saber se o usuário fez as atividades corretamente e também auditar o sistema para saber se o sistema executou todas as operações com sucesso.

Exemplo 2 → Suponha que seu programa realize uma série de configurações no computador antes de poder utilizar o sistema. Seria interessante que todas essas configurações fossem registradas em log para validar qualquer tipo de erro de configuração que aconteça com o uso do aplicativo.

Exemplo 3 → Suponha que você esteja desenvolvendo um programa SIGAD, é obrigatório registrar as operações feitas pelos usuários e as tramitações ocorridas no sistema.

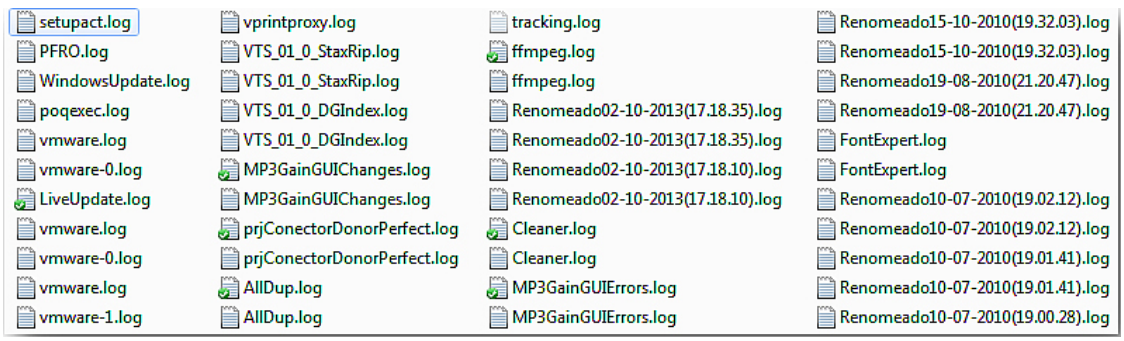
17

5.1 - Onde o log fica armazenado

Geralmente há três locais onde o arquivo de logs de sistemas e aplicativos ficam armazenados, cada uma tem suas particularidades e **benefícios**:

a) **Dentro do banco de dados do sistema** – Muitos programas utilizam banco de dados para guardar informações (um SIGAD precisa de um banco de dados para guardar processos, documentos etc.). Nesse caso, as operações de usuário geralmente são armazenadas em tabelas de log do sistema.
b) **No computador servidor** – Muitos programas precisam realizar operações de acesso à internet, à rede local, a outros servidores ou serviços de TI. Essas funcionalidades geralmente são logadas em arquivos de log no computador servidor da aplicação.
c) **No computador do usuário** – Programas do tipo desktop, que são instalados nas máquinas dos usuários precisam de arquivos de log para registrar informações sobre o uso do programa. Erros como falta de acesso à rede ou Internet, falta de espaço em disco ou memória geralmente são validados pelos programas de computador e logados localmente.

Faça um teste no seu computador. Abra o explorador de arquivo e realize uma pesquisa por "*.log", você ficará impressionado com o tanto de arquivos de log que existe no seu computador.



Exemplo da quantidade de arquivos de log que geralmente encontramos nos nossos computadores.

18

5.2 - O que armazenar em um arquivo de log

Conforme falamos, a ideia é registrar informações que possam ser analisadas e que responda às seguintes perguntas:

✓ **Quais ações o usuário fez?**
✓ **Quais operações o computador realizou?**

Dessa forma, dependendo do objetivo do seu programa, estas abaixo seriam algumas das informações interessantes para serem registradas no log:

- Data de criação do arquivo de log.
- Data de início e término da operação realizada.
- Nome da operação realizada.
- Parâmetros ou conteúdo das variáveis (informações) utilizadas na operação.
- Nome do usuário e/ou login de rede do usuário que realizou a operação.
- Nome e versão do sistema operacional.
- Resultados de testes de conexão.
- Resultados de testes de memória ou processamento.
- Erros encontrados.
- Porta de comunicação utilizada.

E outros.

19

5.3 - Como criar arquivos de log

Arquivos de log são comumente criados da mesma forma que criamos arquivos de texto comum. Cada linha é montada com as informações que queremos registrar no arquivo.

Um detalhe importante é que, geralmente, nunca consultamos nem apagamos o conteúdo de arquivos de log via programação. Os arquivos de log são consultados por editores de texto comuns. O capítulo 3.2 deste módulo apresenta regras gerais de criação de um arquivo de log.

20

6 - Arquivo CSV

Um arquivo **CSV** (Comma Separated Values – Valores Separados por Vírgula), como o próprio nome indica, é uma formatação especial de um arquivo de texto puro onde é possível armazenar dados de uma tabela ou de uma planilha.

Esses dados normalmente referem-se uma sequência de informações semelhantes, uma em cada linha. Se você não é familiarizado com os conceitos de tabela e planilha ainda, imagine que você tenha 5 filhos e para cada um deles você tenha informações de nome, sexo e idade. Uma planilha (ou tabela) que apresentasse essas informações seria, por exemplo:

Nome	Sexo	Idade
João Humberto	Masculino	2 anos
Paula	Feminino	5 anos
Cláudio	Masculino	8 anos
Felipe	Masculino	12 anos
Isadora	Feminino	15 anos.

Um arquivo CSV, por exemplo, pode ser usado temporariamente para trocar dados de um aplicativo com uma planilha (ou tabela) ou vice e versa. Cada linha em um arquivo de Texto CSV representa uma linha em uma planilha. Cada célula (item) em uma linha da planilha geralmente é separada por uma vírgula. No entanto, é possível usar outros caracteres para delimitar um campo, como um caractere tabulador ou o ponto e vírgula.

21

A primeira linha do arquivo CSV pode conter o nome dos campos da planilha. Mas isso não é obrigatório. As demais linhas do arquivo CSV conterão as informações da planilha.

O texto do exemplo acima no formato CSV ficaria assim:

```
Nome,Sexo,Idade
João Humberto,Masculino,2 anos
Paula ,Feminino,5 anos
Cláudio,Masculino,8 anos
Felipe,Masculino,12 anos
Isadora,Feminino,15 anos.
```

Esse arquivo pode ser importado para o Ms Excel, por exemplo, ou para qualquer outro software de planilhas, como o OpenOffice Calc ou o Lotus Notes. Após a importação de informações, novamente você teria cada informação em uma coluna específica, como o exemplo a seguir:

	A	B	C
1	Nome	Sexo	Idade
2	João Humberto	Masculino	2 anos
3	Paula	Feminino	5 anos
4	Cláudio	Masculino	8 anos
5	Felipe	Masculino	12 anos
6	Isadora	Feminino	15 anos.

Após a importação pelo Ms Excel, os dados foram colocados nas mesmas posições originais.

Ao exportar uma planilha para o formato CSV, apenas os dados na planilha atual serão salvos. As demais informações, incluindo fórmulas e formatação, serão perdidas.

22

6.1 - Como criar um arquivo CSV

Vários programas de planilhas e de banco de dados têm a capacidade de criar (exportar) e ler (importar) informações de arquivos CSV. Você também pode criar um programa para fazer essa tarefa. O processo de criação de um arquivo CSV é o mesmo da criação de um arquivo de texto comum (e de um arquivo de log), o que muda é apenas a criação de cada linha, onde precisamos inserir o caractere de vírgula para separar os campos. Vamos fazer um exemplo bem simples de como gravar um arquivo CSV a partir das informações de uma matriz.

```
1  // Passo 1 - Criamos o vetor e adicionamos conteúdo a ele.
2  dim sMatriz [6][3] as string;
3  sMatriz [1][1] = "Nome"; sMatriz [1][2] = "Sexo"; sMatriz [1][3] = "Idade";
4  sMatriz [2][1] = "João"; sMatriz [2][2] = "Masculino"; sMatriz [2][3] = "2 anos";
5  sMatriz [3][1] = "Paula"; sMatriz [3][2] = "Feminino"; sMatriz [3][3] = "5 anos";
6  sMatriz [4][1] = "Cláudio"; sMatriz [4][2] = "Masculino"; sMatriz [4][3] = "8 anos";
7  sMatriz [5][1] = "Felipe"; sMatriz [5][2] = "Masculino"; sMatriz [5][3] = "12 anos";
8  sMatriz [6][1] = "Isadora"; sMatriz [6][2] = "Feminino"; sMatriz [6][3] = "15 anos"
9
10 // Passo 2 - criamos as variáveis e objetos auxiliares
11 // para formatar a linha e inserir o conteúdo no arquivo.
12 dim sTextooArquivo as Archive, i as int, j as int, sLinha as string;
13
14 // Passo 3 - criamos o arquivo vazio.
15 oArquivo.CreateFile ("c:\temp\arquivo_de_teste.csv");
16
17 // Passo 4 - criamos um laço for para criar todas as linhas necessárias
18 for i = 1 to 6 {
19     // Passo 5 - criamos uma string com o conteúdo formatado e separado por vírgulas.
20     sLinha = sMatriz [i][1] + "," + sMatriz [i][2] + "," + sMatriz [i][3]
21     // Passo 6 - inserimos a linha formatada em CSV no arquivo.
22     oArquivo.InputLine (sLinha)}
23
24 // Passo 7 - fechamos e salvamos o arquivo.
25 oArquivo.Close;
```

Exemplo de programa que cria um arquivo CSV.

23

RESUMO

Neste módulo, aprendemos:

a) Que documentos textuais são aqueles que você escreve utilizando programas do tipo “editores de texto”.
b) Que os arquivos de texto são documentos que só contém caracteres (letras, números e símbolos ASCII).

c) Que os arquivos de texto não tem formatação alguma.
d) Que os caracteres CR e LF presentes em arquivos de texto determinam quebras de linha.
e) Que as operações mais comuns de manipulação de arquivos são: ler, gravar, adicionar, criar, renomear e excluir o arquivo.
f) Que arquivos de LOG servem para analisar se determinado procedimento realizou todos os passos corretamente ou para auditar atividades de usuários.
g) Que arquivos CVS servem para troca de dados entre sistemas ou aplicativos diferentes.

UNIDADE 2 – ARQUIVOS DE TEXTO, DOCUMENTOS, PLANILHAS E APRESENTAÇÕES

MÓDULO 2 – ARQUIVOS DE TEXTO – PARTE 2

01

1 - ARQUIVOS DE CONFIGURAÇÃO DO TIPO “INI”.

Olá, seja bem-vindo a mais um módulo de estudo. Já vimos alguns conceitos básicos sobre arquivos de texto e como um programa de computador pode manipular esses arquivos.

Neste módulo, iremos aprender características de outros tipos de arquivos de texto puro (arquivos não binários). Futuramente, quando você aprender a programar em uma determinada linguagem de programação, os conhecimentos adquiridos nesta e nas próximas unidades formarão a base conceitual que você precisará saber para poder manipular arquivos e documentos.

Atenção: para você mesmo fazer testes e verificar os exemplos citados neste módulo, utilizaremos um software livre para edição de arquivos. Como sugestão, baixe e instale o “Notepadd++”, disponível gratuitamente neste endereço: http://notepad-plus-plus.org/.

02

Programas de computador normalmente precisam armazenar informações de configuração. Tanto programas para Windows como para Linux utilizam bastantes arquivos .INI para armazenar configurações. Até mesmo o sistema operacional utiliza arquivos INI para guardar algumas configurações.

As informações mais comuns que vemos em **arquivos INI** são:

- Informações de registro do programa, como nome do usuário, número de série, data de registro, e outros;
- Informações do hardware como tamanho do vídeo, cores, fontes para serem utilizadas, tamanho e posição de janelas etc.
- Versão do programa, muitas vezes utilizada para controlar a atualização do próprio sistema;
- Se o programa se conecta a algum servidor ou banco de dados, informação desses acessos;
- Configurações, informações preferenciais e campos formatados para o usuário etc.;
- Portas de acesso de internet, quantidade de conexões, velocidade etc., essas informações também podem estar em arquivos INI.

03

1.1 - Estrutura de um arquivo INI

Um arquivo INI geralmente tem a seguinte estrutura:
• Linhas de comentário, iniciadas por ponto e vírgula (Exemplo: "; Exemplo de um arquivo .INI");
• Título da seção, para nomear um conjunto de funcionalidades/configurações, escrito entre colchetes (Exemplo: "[Banco de dados]");
• Linhas de configuração com as seguintes informações:
o O item de configuração, que representa o item a ser configurado (Exemplo: "IP_Servidor";
o O sinal de igual ("=");
o Valor da configuração, que geralmente é um texto, um número ou uma data (Exemplo: "100.10.0.118").

Vamos ver um exemplo do conteúdo de um arquivo .INI:

```
1  ; Exemplo de um arquivo .INI
2  ; Esta linha é uma linha de comentários.
3  [Banco de Dados]
4  IP_Servidor=100.10.0.118
5  Base_Dados=sistema_RH
6
7  [Usuario]
8  Nome_Usuario=Joao.Carlos
9  Ultimo_acesso=10/12/2013
```

• **Nas linhas 1 e 2** vemos que elas iniciam com ";" e por isso são apenas comentários;
• **A linha 3** possui a palavra "Banco de Dados" entre colchetes, então, o nome desse rótulo é "Banco de dados", as duas informações das linhas 4 e 5 pertencem ao rótulo "Banco de Dados";
• **As linhas 4 e 5** apresentam duas configurações:
• **A linha 4** diz que a variável "Ip_Servidor" tem o valor 100.10.0.118
• **A linha 5** diz que a variável "Base_Dados" tem o valor "sistema_RH".
• **Na linha 7**, um novo rótulo é apresentado, o rótulo "Usuario".
• **Nas linhas 8 e 9**, duas novas variáveis são configuradas.

04

1.2 - Como ler e gravar informação em um arquivo .INI

Para acessar arquivos INI não é necessário ler e interpretar o conteúdo do arquivo linha a linha. Por isso não devemos utilizar os comandos básicos de edição de arquivo de texto que explicamos no módulo anterior. O sistema operacional e as linguagens de programação oferecem funções específicas para realizar as operações de leitura e gravação em arquivo INI, essas funções estão dentro de um conjunto maior de funções chamado API do Windows. As duas funções utilizadas para ler e gravar conteúdo em arquivos INI são respectivamente**GetPrivateProfileString** e **SetPrivateProfileString**.

Ao invés de ler todo o arquivo e analisar linha a linha, por meio das funções da API do Windows, seu

programa de computador só precisa executar um único procedimento para ler ou gravar informações de um arquivo INI.

Ainda, se a linguagem de programação for bem moderna, provavelmente ela incluirá funcionalidades semelhantes a estas, que também tornará seu trabalho de manipulação muito fácil.

Vamos ver um exemplo para ficar mais fácil entender. Vamos considerar que o arquivo de configuração chama-se "config.ini" e que ele está localizado na pasta "c:\programa". Vamos considerar que o conteúdo dele é o apresentado no item 1.1:

```
1  ; Exemplo de um arquivo .INI
2  ; Esta linha é uma linha de comentários.
3  [Banco de Dados]
4  IP_Servidor=100.10.0.118
5  Base_Dados=sistema_RH
6
7  [Usuario]
8  Nome_Usuario=Joao.Carlos
9  Ultimo_acesso=10/12/2013
```

Arquivo exemplo: "c:\programa\config.ini"

Para ler o valor do IP do servidor do arquivo INI usaríamos, hipoteticamente, o seguinte comando:

```
GetPrivateProfileString("Banco de dados", "IP_Servidor", "", sIP_Servidor, 0, "c:\programa\config.ini")
```

05

Vamos entender como esse comando funciona:

- **GetPrivateProfileString** → É o nome da função do Windows que lê dados de um arquivo INI, essa é a função que estamos executando.
- **Banco de dados** → É o título da seção onde está o valor da configuração que queremos ler. Observe que no arquivo INI, o título está escrito entre colchetes: [Banco de dados].
- **IP_Servidor** → É a configuração que queremos ler. A função irá ler o texto que estiver do lado direito do sinal de igual ("=").
- **sIP_Servidor** → É a variável texto que nosso programa receberá com o conteúdo assim que a função for executada. No caso, após a execução da função, ela receberá o valor "100.10.0.118".
- **c:\programa\config.ini** → É o nome e o endereço completo de onde o arquivo de configuração está localizado dentro do computador.

Antes da execução da função, o valor da variável sIP_Servidor é vazio, não contém nada. Após a execução da função, essa variável irá conter o valor “100.10.0.118”. Daí o programador faz o uso que ele quiser com essa informação, como por exemplo apontar o programa para esse servidor de banco de dados.

Para ler a data de último acesso do usuário ao sistema, o programador executaria o seguinte procedimento (bem semelhante ao anterior):

```
GetPrivateProfileString("Usuario", "Ultimo_acesso", "", dUltimoAcesso, 0, "c:\programa\config.ini")
```

Veja que interessante, não foi preciso abrir o arquivo e ler linha a linha até encontrar a linha desejada e depois interpretar cada linha para saber se é aquele que interessa ao programador, a função GetPrivateProfileString faz tudo isso para ele. Basta um único comando e o sistema operacional faz tudo o que é necessário para ler o conteúdo correto do arquivo para o programador.

06

Para gravar informação em um arquivo INI é bem fácil também, e o comando é muito parecido, é o SetPrivateProfileString.

Vamos gravar a data de “17/10/2014” como data de último acesso ao sistema. O comando seria o seguinte:

```
SetPrivateProfileString("Usuario", "Ultimo_acesso", "", "17/10/2014", 0, "c:\programa\config.ini")
```

Observações importantes:

1- Os exemplos acima não têm por objetivo apresentar todo conhecimento necessário a um programador para ler ou gravar informações em arquivo INI, mas tão somente objetiva mostrar ao usuário que há formas mais fáceis e melhor controladas para tratar esse tipo de arquivo. Quando você estudar as matérias de linguagem de programação você aprenderá a realizar essas operações com todos os detalhes necessários.

2- Os sistemas operacionais da família Windows possuem outras técnicas para armazenar informações de configuração de sistemas. A mais comum é o uso da Register, que é uma espécie de banco de dados de configurações. O sistema operacional também possui comandos para acesso direto à register. Para conhecer um pouco mais sobre esses comandos, pesquise e, sites de busca por “DeleteSetting, GetAllSettings, GetSetting and SaveSetting”.

07

2 - ARQUIVOS XML (EXTENSIBLE MARKUP LANGUAGE)

Arquivos XML são arquivos de texto com marcação. Assim com o HTML, o XML utiliza uma técnica próxima de elaboração e interpretação.

XML traz uma sintaxe básica que pode ser utilizada para compartilhar informações entre diferentes computadores e aplicações. Quando combinado com outros padrões, torna possível definir o conteúdo de um documento separadamente de seu formato, tornando simples para reutilizar o código em outras aplicações para diferentes propósitos.

Portanto, uma das suas principais características é sua **portabilidade**, pois, por exemplo, um banco de dados pode escrever um arquivo XML para que outro banco consiga lê-lo.

A seguir serão listadas outras características dos arquivos XML, tais como seus principais usos, os formatos, leitura e gravação desse tipo de arquivo etc.

08

2.1 - Principais usos de arquivos XML

Arquivos XML têm os mais variados propósitos, os mais comuns são:

- Troca de informações entre sistemas;
- Registro de log (com o mesmo objetivo dos arquivos LOG);
- Configuração de sistemas (com o mesmo objetivo dos arquivos INI);
- Documentos ou planilhas em formato universal;
- Registro de dados (pequeno banco de dados).

2.2 - Quando devemos usar XML, LOG ou INI

A melhor resposta é: **depende só de você ou do seu projeto**. A tecnologia XML é mais moderna que a utilizada em arquivos LOG e INI, entretanto, para o usuário comum, é muito mais difícil ler e interpretar o conteúdo de um arquivo XML do que de um arquivo LOG ou INI. A equipe do seu projeto pode definir se uma ou outra tecnologia será usada.

09

2.3 - Formato do arquivo XML

O símbolo comumente utilizado para distinguir as marcações do conteúdo são os símbolos de “<” e “>”. O padrão de escrita de uma marcação é:

```
<nome do marcador>
  <Variável = “conteúdo”/>
</nome do marcador>
```

Exemplo:

```
<Pessoa>
  <Nome = “Pedro Xavier Carvalho”/>
</Pessoa>
```

Normalmente, os arquivos XML começam com uma linha que identifica o padrão de codificação do arquivo. Essa linha geralmente é descrita assim:
`<?xml version="1.0" encoding="utf-8"?>`

Essa informação acima diz que o arquivo é compatível com qualquer versão de interpretador XML 1.0 ou superior e que utiliza o padrão utf-8 para caracteres especiais (forma pela qual se registram letras acentuadas em diversos idiomas, como a letra “ã” ou a letra espanhola “ñ”).

Após essa linha, todo o restante é o conteúdo propriamente dito do arquivo. Arquivos XML são comumente utilizados para armazenar informação estruturada do tipo 1-n. Esse tipo de informação é aquela que temos um título (assunto) e diversas outras informações sobre aquele título.

10

Exemplo de um arquivo xml utilizado com o propósito de log de operação (semelhante a um arquivo de log):

teste.xml

```
<?xml version='1.0' encoding='utf-8'?>
<Registro_de_log>
<Identificacao>
    <id>30403338_1356076770</id>
    <cliente>Roberto Araujo Gomes</cliente>
    <operacao>pagamento via internet</operacao>
    <detalhe>pagamento de boleto simples</detalhe>
    <data_hora>10/10/2014 14:54:38</data_hora>
</Identificacao>
    <Operacoes>
        <Operacao>1. Validar dados cliente, resultado ok</Operacao>
        <Operacao>2. Validar dados bancários, resultado ok</Operacao>
        <Operacao>3. Validar boleto, resultado ok</Operacao>
        <Operacao>4. Validar saldo em conta, resultado ok</Operacao>
        <Operacao>5. Enviar boleto, resultado ok</Operacao>
        <Operacao>6. Processar boleto, resultado ok</Operacao>
        <Operacao>7. Pagamento efetivado com sucesso, resultado ok</Operacao>
    </Operacoes>
</Registro_de_log>
```

Apesar das marcações, podemos identificar várias informações, como o nome do cliente, a data da operação, e o passo a passo das operações realizadas.

11

2.4 - Como ler e gravar arquivos XML

O sistema operacional também oferece meios práticos para ler e gravar arquivos XML, assim como já comentamos para arquivos .INI. O componente do sistema operacional utilizado para tratar arquivos XML é chamado de DOM Documento Builder. Esse componente implementa todas as operações necessárias para ler, gravar e alterar informações de arquivo XML.

2.4.1 - Exemplo em Java

A título de ilustração, vamos ver quais seriam os comandos em Java para ler e tratar o conteúdo de um arquivo XML.

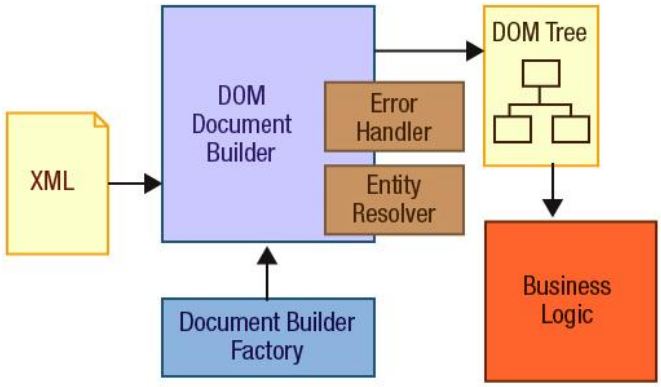
Para este exemplo, vamos utilizar este XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<agenda>
      <contato id="01">
            <nome>Maria da Silva</nome>
            <endereco>Rua Ficticia, 32, Bairro Ficticio,
Ficticia City</endereco>
            <telefone>21 - 32658978</telefone>
            <email>maria.silva@provedor.com.br</email>
      </contato>
      <contato id="02">
            <nome>João da Silva</nome>
            <endereco>Rua Inventada, 12, Bairro Inventado,
Inventada City</endereco>
            <telefone>11 - 24569865</telefone>
            <email>joao.silva@provedor.com.br</email>
      </contato>
</agenda>
```

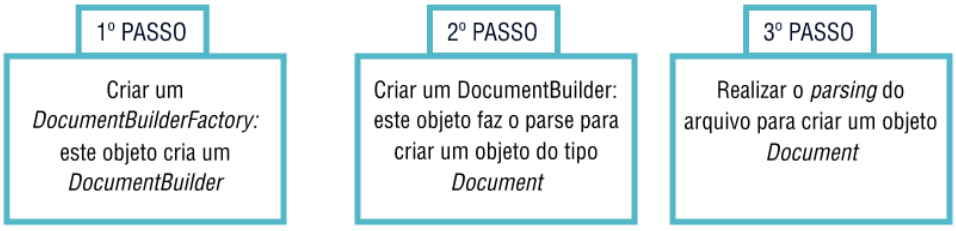
12

Para trabalhar com a informação de um arquivo XML, é necessário primeiro criar um objeto ***Document***. Para gerenciar o objeto *Document*, precisaremos de um objeto ***Factory***. O processo exato pode variar de implementação para implementação, mas as ideias são as mesmas.

A figura abaixo exemplifica o processo de criar um XML com a API DOM (fonte: SUN):



Os três **passos** em um ambiente java são:



Uma vez que o é feito o parse do arquivo (criação e instanciação) e o objeto Document é criado, a aplicação pode explorar a estrutura da árvore XML, exibindo a informação na tela, guardando os dados em um objeto, entre outras operações.

13

Todo documento XML inicia-se pelo elemento raiz. Um documento XML bem formado possui apenas um elemento raiz, também conhecido como DocumentElement. Então, o primeiro passo é a nossa aplicação obter esse elemento.

```
1   //Passo 1: obter o elemento raiz
2   Element raiz = doc.getDocumentElement();
3   System.out.println(&amp;quot;O elemento raiz é: &amp;quot; + raiz.getNodeName());
```

Após a determinação do *element* raiz, podemos obter a lista de elementos **filhos**. O DOM fornece um tipo especial de interface de coleção chamada *NodeList* usada para referenciar uma lista de referências do tipo Node. Ou seja, sempre que desejarmos obter os filhos de um elemento, devemos utilizar esse método, mesmo sabendo que este determinado elemento possui apenas 1 sub-elemento.

Vamos obter agora os elementos do tipo **contato**:

```
5   //Passo 2: localizar os elementos filhos da agenda
6   NodeList listaContatos = raiz.getElementsByTagName(&amp;quot;contato&amp;quot;);
```

Como temos uma coleção de elementos contato, vamos percorrer cada um deles para obter as informações que queremos:

```
8   //Passo 3: obter os elementos de cada elemento contato
9   for (int i=0; i&amp;lt;listaContatos.getLength(); i++){
10
11  //como cada elemento do NodeList é um nó, precisamos fazer o cast
12  Element contato = (Element) listaContatos.item(i);}
```

14

Agora que já temos acesso ao elemento contato, vamos começar a extrair as informações. Nosso primeiro passo é obter o valor do atributo *id*:

```
14  //Passo 4: obter o atributo id do contato
15  Attr id = contato.getAttributeNode(&amp;quot;id&amp;quot;);
16  System.out.println(&amp;quot;Contato id: &amp;quot; + id.getNodeValue());
```

Após obtermos o atributo, vamos começar a extrair as informações dos elementos do contato. Para isso, vamos obter um NodeList com os elementos do tipo 'nome'. Como sabemos que o arquivo XML tem apenas um elemento do tipo nome para cada elemento do tipo contato, podemos fazer referência diretamente para o primeiro elemento dessa lista. Um nó do tipo elemento possui valor *null*, então não

podemos pegar diretamente o valor desse nó, temos que pegar o valor do nó texto, que obtemos através do método *getFirstChild()*:

```
18  //Passo 5: obtém o nome do contato
19  NodeList listaNomes = contato.getElementsByTagName(&amp;quot;nome&amp;quot;);
20  Node nome = listaNomes.item(0).getFirstChild();
21  System.out.println(&amp;quot;Nome: &amp;quot; + nome.getNodeValue());
```

15

Fazemos isso para cada informação que desejamos extrair: endereço, telefone e e-mail. Apenas precisamos modificar a String passada como argumento pelo método *getElementsByTagName*. Fica assim:

```
23  //Passo 6: obtém o endereço do contato
24  NodeList listaEndereco = contato.getElementsByTagName(&amp;quot;endereco&amp;quot;);
25  Node endereco = listaEndereco.item(0).getFirstChild();
26  System.out.println(&amp;quot;Endereço: &amp;quot; + endereco.getNodeValue());
27
28  //Passo 7: obtém o telefone do contato
29  NodeList listaTelefone = contato.getElementsByTagName(&amp;quot;telefone&amp;quot;);
30  Node telefone = listaTelefone.item(0).getFirstChild();
31  System.out.println(&amp;quot;Telefone: &amp;quot; + telefone.getNodeValue());
32
33  //Passo 8: obtém o email do contato
34  NodeList listaEmail = contato.getElementsByTagName(&amp;quot;email&amp;quot;);
35  Node email = listaEmail.item(0).getFirstChild();
36  System.out.println(&amp;quot;Email: &amp;quot; + email.getNodeValue());
```

E acabamos de extrair todas as informações do XML!

16

O código de parsing também precisa considerar a captura de erros de exceção para reportar os vários erros que podem acontecer, como por exemplo:

- ParserConfigurationException;
- SaxParseException;
- IOException.

Todo o código acima está dentro de um bloco *try-catch*:

```
38  } catch (ParserConfigurationException e) {
39  System.out.println(&amp;quot;O parser não foi configurado corretamente.&amp;quot;);
40  e.printStackTrace();
41  } catch (SAXException e) {
42  System.out.println(&amp;quot;Problema ao fazer o parse do arquivo.&amp;quot;);
43  e.printStackTrace();
44  } catch (IOException e) {
45  System.out.println(&amp;quot;O arquivo não pode ser lido.&amp;quot;);
46  e.printStackTrace();
47  }
```

Podemos melhorar o código na hora de obter as informações do contato, criando um método genérico para extrair o valor de uma *tag* específica:

```
49  public String obterValorElemento(Element elemento, String nomeElemento){
50    //obtém a lista de elementos
51    NodeList listaElemento = elemento.getElementsByTagName(nomeElemento);
52  if (listaElemento == null){
53    return null;
54    }
55    //obtém o elemento
56    Element noElemento = (Element) listaElemento.item(0);
57  if (noElemento == null){
58    return null;
59    }
60    //obtém o nó com a informação
61    Node no = noElemento.getFirstChild();
62    return no.getNodeValue();
63    }
```

ParserConfigurationException
Acionado quando o *DocumentBuilderFactory* não consegue criar o parser.

SaxParseException
Ocorrendo quando o parser encontra um problema na formatação do arquivo XML. O objeto *Exception* carrega informações sobre a localização do erro no arquivo.

IOException
Acionado quando um erro de arquivo ocorre.

17

Podemos melhorar mais ainda construindo um código que retorne uma lista de objetos do tipo Contato. Para isso, precisamos também construir uma classe do tipo Contato:

```
65  public class Contato {
66
67    private int id;
68    private String nome;
69    private String endereco;
70    private String telefone;
71    private String email;
72
73    //métodos getters e setters
74    }
```

O método que cria um novo contato:

```
76  public Contato criaContato(Element elemento){
77    Contato contato = new Contato();
78    contato.setId(Integer.parseInt(elemento.getAttributeNode(&amp;quot;id&amp;quot;).getNodeValue()));
79    contato.setNome(obterValorElemento(elemento,&amp;quot;nome&amp;quot;));
80    contato.setEndereco(obterValorElemento(elemento,&amp;quot;endereco&amp;quot;));
81    contato.setTelefone(obterValorElemento(elemento,&amp;quot;telefone&amp;quot;));
82    contato.setEmail(obterValorElemento(elemento,&amp;quot;email&amp;quot;));
83    return contato;
84    }
```

E nosso novo código:

```
86    //Passo 3: obter os elementos de cada elemento contato
87  for (int i=0; i&amp;lt;listaContatos.getLength(); i++){
88
89    //como cada elemento do NodeList é um nó, precisamos fazer o cast
90    Element elementoContato = (Element) listaContatos.item(i);
91
92    //cria um objeto Contato com as informações do elemento contato
93    Contato contato = criaContato(elementoContato);
94    System.out.println(contato);
95    }
```

18

2.5 - Exemplo mais moderno em Visual Basic dot Net

Em Visual Basic dot Net, a classe *XmlTextReader* permite a leitura de informações de um arquivo XML. Essa classe fornece a análise direta e a simbolização completa compatível com o XML 1.0. A classe *XmlTextReader* é uma evolução melhorada, mais rápida e mais simples de utilizar do que a classe DOM (que utilizamos no exemplo Java).

Vamos ver como fica o código fonte em Visual Basic de um exemplo hipotético para a leitura de um documento XML. Analise principalmente como é muito mais simples que o código DOM do exemplo em Java.

```
' Passo 1, dizer para o programa que utilizaremos XMLTextReader
Imports System.Xml

' Passo 2: criar um objeto reader e faz ele abrir o arquivo "arquivo.xml"
Dim reader As XmlTextReader = New XmlTextReader ("arquivo.xml")  '

' Passo 3: Ler todo o conteúdo do arquivo XML
        Do While (reader.Read())
            Select Case reader.NodeType
                Case XmlNodeType.Element 'Exibir o início do elemento.
                    Console.Write("<" + reader.Name)
                    If reader.HasAttributes Then 'Se existirem atributos
                        While reader.MoveToNextAttribute()
                            'Exibir o nome e o valor do atributo.
                            Console.Write(" {0}='{1}'", reader.Name, reader.Value)
                        End While
                    End If
                    Console.WriteLine(">")
                Case XmlNodeType.Text 'Exibir o texto em cada elemento.
                    Console.WriteLine(reader.Value)
                Case XmlNodeType.EndElement 'Exibir o fim do elemento.
                    Console.Write("</" + reader.Name)
                    Console.WriteLine(">")
            End Select
        Loop
' Leitura do arquivo XML concluída. Fim!
```

Nossa! Como nosso código fonte ficou mais simples! Gastamos apenas 25% (ou ¼) do número de linhas do exemplo em Java!

A evolução das linguagens de programação fazem isso dia a dia: tornam o trabalho do programador cada vez mais poderoso, de maneira mais simples, rápida e segura. Espere sempre novas formas mais inteligentes de fazer o mesmo trabalho.

19

3 - ARQUIVOS DE SCRIPT

Arquivos de script são como miniprogramas: executam atividades pré-concebidas de forma automatizada. Há uma variedade muito grande de scripts, dentre os principais tipos, destacamos:

- **Scripts de páginas web**, como o JavaScript.
- **Scripts de sistema operacional**, como os arquivos em lote (Bat) do Ms DOS e da família Windows, e os scripts Unix e Linux (shell script).
- **Scripts de banco de dados**, como a linguagem Ansi SQL.

Arquivos de script são arquivos de texto puro, assim como os arquivos TXT. Um programa de computador dificilmente manipulará arquivos de script, mas é importante você saber como scripts podem auxiliar sua vida de programador.

20

3.1 - JavaScript

De todos os tipos de arquivo de script, provavelmente o único que você terá maior contato será o JavaScript. O JavaScript é tão poderoso que é por si só considerado como sendo uma linguagem de programação. O JavaScript funciona apenas em páginas WEB dentro do código HTML dessas páginas.

Uma página HTML possui várias TAGs, as TAGs específicas para o JavaSript são < script > e < /script >. Tudo o que estiver dentro dessas TAGs será considerados código JavaScript.

A ideia do JavaScript é a seguinte: em sistemas WEB todas as operações são feitas no computador servidor que executa o sistema. Então, em um ambiente que não existe (ou não foi implementado o JavaScript) todas as validações de campos da tela tem que ocorrer no servidor. Com o JavaScript, diversos processamentos, testes e operações podem ser feitas no computador do usuário, assim, há uma grande economia de processamento no servidor, economia de banda de rede e aumento de velocidade do sistema de modo geral.

A seguir veremos um exemplo para explicar melhor a ideia.

21

Exemplo

Suponha que o seu sistema realize o cadastro de um novo cliente no sistema. Suponha que existam os seguintes campos obrigatórios: nome, endereço, CEP, cidade, UF, CPF e e-mail; suponha também que existam os seguintes campos opcionais: telefone residencial e telefone celular.

Em um ambiente sem o JavaScript, o programa precisa receber todas as informações do usuário, validar se os campos obrigatórios foram preenchidos, se além de preenchidos obedecem a regras e formatos esperados (como a o formato do CPF e a regra de cálculo do dígito verificador, o formato do CPF, o formato dos campos de telefone). Se algo der errado, o sistema deve informar que não pode fazer o cadastro porque tal informação está faltando ou preenchida de forma errada. Para esse exemplo bem simples, temos os seguintes testes a serem feitos:

- Sete testes de campos obrigatórios preenchidos;
- Oito testes de formatos de campos esperados;
- Um teste de dígito verificador.
- Total: 16 testes!

Nesse cenário, há 16 chances de erro no preenchimento do formulário e por 16 vezes o sistema pode validar os dados e impedir o cadastramento. São 16 processamentos “desperdiçados” no servidor! Agora, imagine centenas de cliente fazendo cadastros, olha como o processamento do servidor ficaria extremamente onerado somente fazendo validações de campos do formulário! Isso é um desperdício!

O JavaScript surgiu para resolver esse problema: ao invés de fazer esses testes no computador servidor, ele faz os testes no computador o usuário! Muito prático, correto? Ao invés de onerar o servidor, o próprio computador do usuário realiza os testes. Poupamos não só o processamento do servidor, mas

também o próprio uso da banda de Internet. Somente quando os 16 testes passassem no computador do usuário é que o sistema enviaria os dados (já certinhos) para serem cadastrados no banco de dados. Que ótima ideia!

O JavaScript é inserido dentro do código HTML das páginas WEB do seu programa.

22

3.1.1 - Utilidade do JavaScript

Hoje, portanto, a grande maioria dos sistemas implementam rotinas que fazem pequenas operações, como esses testes, na máquina dos clientes. Alguns **exemplos de uso**:

- Criação de menus de funcionalidades de programas;
- Mudanças de layout de tela;
- Preenchimento automático de campos;
- Manipulação da janela, como o controle programático sobre seu tamanho, posição e atributos;
- Validação de um formulário para garantir que são aceitáveis antes de serem enviados ao servidor;
- Manipulação de imagens, à medida que o mouse se movimenta sob elas.

23

3.1.2 - Características tecnológicas do JavaScript

Algumas das **características** do JavaScript importantes de serem ressaltadas são:

- O JavaScript suporta os elementos de sintaxe de **programação estruturada** da linguagem C (por exemplo, if, while, switch).
- **Tipagem dinâmica**: no JavaScript os tipos são associados com valores, não com variáveis. Por exemplo, a variável x poderia ser associada a um número e mais tarde associada a uma string.
- É quase inteiramente **baseada em objetos**: propriedades e seus valores podem ser adicionadas, mudadas, ou deletadas em tempo de execução. A maioria das propriedades de um objeto pode ser enumerada usando-se uma estrutura de repetição for...in.
- JavaScript inclui a **função eval** que consegue executar em tempo de execução comandos da linguagem que estejam escritos em uma string.
- Por ser tão popular, todos os programas navegadores de Internet suportam JavaScript.

24

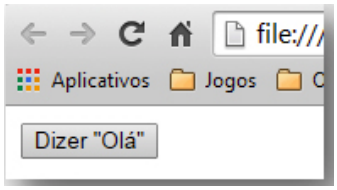
3.1.3 - Exemplo de programa em JavaScript

Um exemplo minimalista de uma página conforme os padrões web que contém JavaScript pode ser representado pelo seguinte código (a parte JavaScript está destacada em cinza entre as linhas 6 a 12, as outras linhas referem-se à informações HTML):

teste_javascript.html

```
<!DOCTYPE html>
<html lang="pt-BR">
    <head>
        <meta charset="UTF-8" />
        <title>Wikipédia</title>
        <script>
            window.onload = function () {
                document.getElementById("hello").addEventListener("click", function () {
                    alert("Bem-vindo ao JavaScript!");
                }, false);
            };
        </script>
    </head>
    <body>
        <noscript>Seu navegador não suporta JavaScript ou ele está desabilitado.</noscript>
        <button id="hello">Dizer "Olá"</button>
    </body>
</html>
```

Ao executar esse arquivo, aparecerá o seguinte conteúdo na tela do seu navegador:



25

3.2 - Script de linhas de commando

Scripts de linha de comando são miniprogramas que executam tarefas ligadas ao sistema operacional e seus serviços (como banco de dados, servidores de e-mail, diretório de usuários etc.).

Esses scripts são muito utilizados por profissionais que administram redes de dados, pois os scripts permitem realizar uma série de miniprogramas de gerenciamento de rede. Entre as **necessidades que podem ser solucionadas por scripts de comando**, as mais importantes seriam:

- Realização de backup de arquivos.
- Mostrar informações do sistema operacional e da rede de dados.
- Execução de operações de teste de rede.
- Cadastro de usuários no domínio da rede.
- Iniciação ou parada de serviços de rede (como os servidores de aplicação, de e-mail ou de segurança).
- Instalar ou desinstalar programas.
- Criar usuários e alterar senhas.
- Manipular bancos de dados (exemplo: comandos em Linux para manipular o MySQL).
- Replicação de configurações em diversos computadores.
- Copiar arquivos de uma pasta para outra.
- Verificar se a Internet está ativa e funcional.

- Cadastrar um usuário na rede do Windows.
- Instalar um programa no sistema Linux.
- Compilar um programa em uma determinada linguagem de programação.

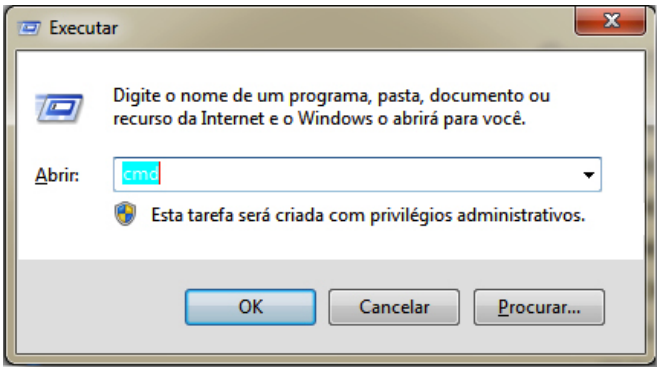
No ambiente Microsoft há dois tipos de scripts: comandos de linha básico (comandos batch) e visual basic script, conforme veremos a seguir.

26

3.2.1 - Comandos batch

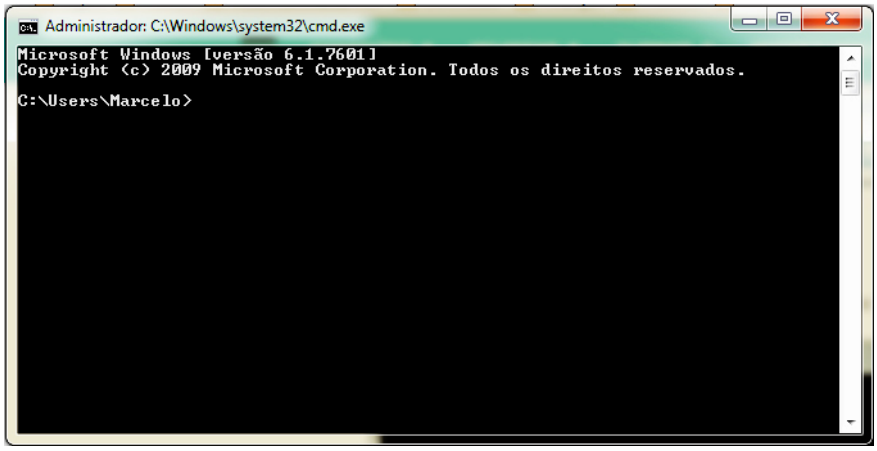
Os comandos batch são miniprogramas que executam operações do sistema operacional apenas. Esses comandos realizam testes de rede, fazem manipulação de arquivos (cópias, backups, exclusões etc.).

São muito simples e a interface tem que ser a linha de comando do sistema operacional, também chamada de console ou shell, que é aquela tela preta dos sistemas antigos. Para abrir a console do Windows, mantenha pressionada a tecla de bandeira do Windows e depois pressione R, na janela que abrir, digite CMD e pressione ENTER.



27

Ao pressionar a tecla da bandeira do Windows e a letra R, aparece a tela de “Executar”.



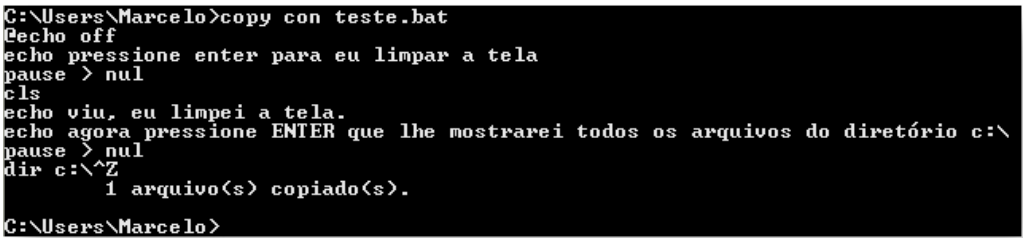
Ao digitar CMD e Enter, aparece a console do Windows.

28

Vamos criar agora um programa batch bem simples. Vamos criá-lo diretamente na console (o que não é uma boa ideia, melhor seria usar um editor de texto para isso). A partir da tela de console, digite os seguintes comandos abaixo pressionando Enter ao final de cada linha:

```
copy con teste.bat
@echo off
echo pressione enter para eu limpar a tela
pause > nul
cls
echo viu, eu limpei a tela.
echo agora pressione ENTER que lhe mostrarei todos os arquivos do diretório c:\
pause > nul
dir c:\
```

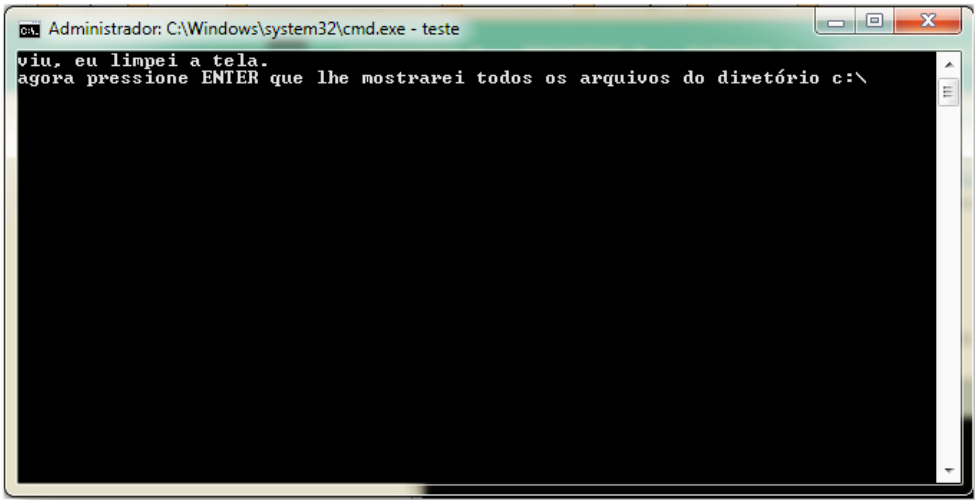
Na última linha, após digitar o texto “dir c:\”, pressione a tecla F6 para gravar o arquivo. Se tudo deu certo, na tela do seu computador deve ter aparecido algo muito próximo ao da tela abaixo:



29

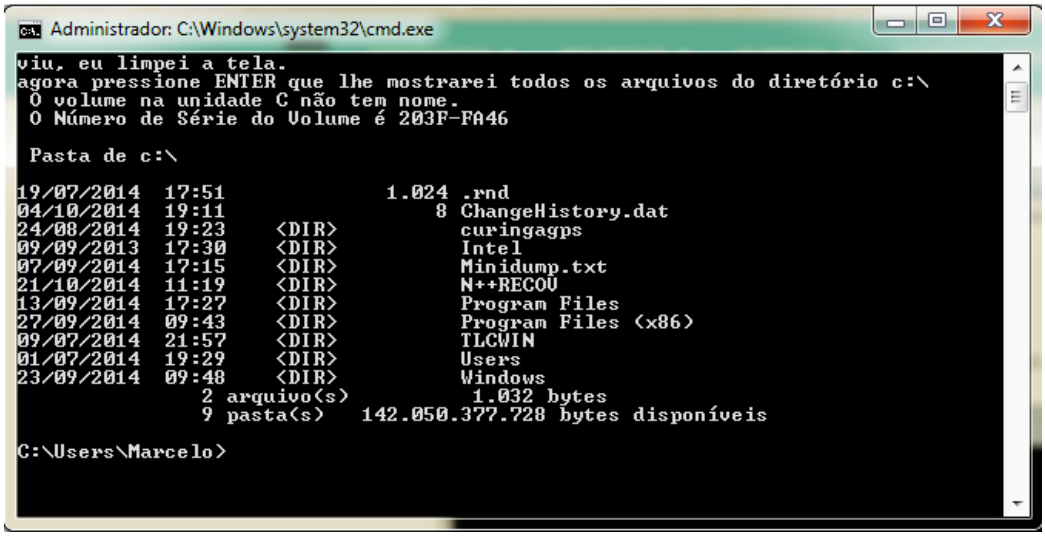
Agora é hora de testar o seu programa, na tela de console, digite “teste” e pressione ENTER. O resultado esperado é algo parecido com a tela abaixo:

Pressione Enter agora. Sua tela deve aparecer como esta aqui:



30

Pressione Enter novamente, agora em sua tela deve aparecer algo semelhante à tela abaixo:



Viu? Você criou um programa batch que limpa a tela e mostra a listagem do diretório raiz.

Para sair da console, digite EXIT e pressione ENTER.

31

Os programas em batch geralmente são criados com editores de texto simples, como o Notepad. Todos os sistemas operacionais permitem a criação de miniprogramas batch. Criar diretamente na console não é indicado, pois se você errar o texto digitado, a console não permite que você conserte o erro, assim você terá que digitar tudo novamente.

Leia mais em:

• Introdução a arquivos .BAT e programação em loteshttp://www.devmedia.com.br/introducao-a-arquivos-bat-e-programacao-em-lotes/24800#ixzz3GpBorukt

Fazendo mágica com batch files http://www.superdownloads.com.br/materias/5887-fazendo-magica-batch-files.htm

O Unix e o Linux também permitem a criação de scripts. Nesses sistemas operacionais, ao invés de o arquivo possuir a extensão BAT ele geralmente possui a extensão SH.

Para saber mais sobre scripts Unix e Linux, acesse:
• http://www.devmedia.com.br/introducao-ao-shell-script-no-linux/25778
• http://www.linuxdescomplicado.com.br/2013/10/saiba-como-criar-uma-interface-grafica.html
• http://www.nacaolivre.com.br/shell-script/criando-programas-shell-script/
• http://www.hardware.com.br/livros/linux/uma-introducao-shell-script.html

Como programador, você utilizará muito pouco de scripts de sistema operacional. Mas é importante que você saiba o que são e para que servem, pois os administradores da rede da empresa que você trabalha podem criar scripts para gerenciar os programas que você fizer.

32

3.3 - Scripts de banco de dados

Scripts de banco de dados utilizam a linguagem SQL e servem geralmente para:

- Criar e alterar bancos de dados, tabelas, índices, relacionamentos, usuários e permissões.
- Fazer carga ou extração de dados de banco de dados;
- Analisar dados cadastrados no banco de dados.

Para executar essas operações em bancos de dados você precisa conhecer os comandos de manipulação de dados. Os principais comandos de manipulação são:

Função	Comandos SQL	Descrição do comando	Exemplo
Criações	CREATE	Cria um bando de dados, uma visão ou uma tabela.	*Create database RH;* Cria um banco de dados chamado RH. *Create table Pessoa (id as integer, nome as varchar, sexo as varchar);* Criar uma tabela de nome Pessoa com os campos ID, Nome e Sexo.
Inclusões	INSERT	Insere registros (dados) em uma tabela existente.	*Insert into Pessoa (id, nome, sexo) values (1,"Marcelo","Masculino");* Insere o Registro "1, Marcelo, Masculino", dentro da tabela Pessoa.
Consultas	SELECT	Realiza consultas de dados pertencentes a uma tabela ou mais tabelas.	*Select * From Pessoa;* Mostra todos os dados da tabela Pessoa.
Alterações	UPDATE	Altera os valores de dados em uma ou mais linhas de uma tabela existente.	*Update Pessoa set nome = 'Thiago' where id_pessoa = 1;* Altera o nome do registro de ID 1 (de Marcelo) para Thiago da tabela Pessoa.
Exclusões	DELETE	Remove linhas (registros) existentes de uma tabela.	*Delete from Pessoa where id_pessoa = 1* Apaga o registro de ID 1 da tabela pessoa.
Exclusões	DROP	Apaga uma tabela ou exclui um bando de dados do Sistema.	*Drop table pessoa;* Exclui a tabela pessoa do banco de dados. *Drop database RH;* Remove o banco de dados RH do sistema.

33

Quando você estudar a matéria de banco de dados, você verá a fundo como esses comandos funcionam. Aqui vamos deixar apenas esse pequeno registro para você ter uma ideia da sintaxe desses comandos.

Para saber mais, consulte:
- http://pt.wikipedia.org/wiki/SQL
- http://pt.slideshare.net/reuellopes/apostila-introduo-linguagem-sql
- http://www.dicasdeprogramacao.com.br/o-que-e-sql/
- http://pgdocptbr.sourceforge.net/pg80/tutorial-sql.html

Como programador, você utilizará bastante de scripts de banco de dados e comandos SQL. Eles são essenciais para qualquer tipo de programa que armazene dados dentro um banco de dados. Um script de banco de dados pode, por exemplo, criar o banco de dados da sua aplicação, criar todas as tabelas, índices e relacionamentos, popular as tabelas de dados de referência.

34

Resumo

Neste módulo, aprendemos:

a) Que arquivos INI são utilizados para configuração de sistemas.
b) Que o sistema operacional e as linguagens de programação oferecem meios de facilmente manipular arquivos INI.
c) Que os componentes de um arquivo INI são linhas de comentários, título da seção, item de configuração e o valor da configuração.
d) Que arquivos XML são utilizados para armazenamento e troca de informações.
e) Que o JavaScript é tão poderoso que é considerada uma linguagem de programação.
f) Que o JavaScript funciona dentro das páginas HTML entre as TAGs < script > e < /script >.
g) Que scripts de linha de comando são pequenos programas que executam operações em sistemas operacionais e seus serviços.
h) Que scripts do tipo comandos batch funcionam em sistemas operacionais apenas.
i) Que scripts de banco de dados funcionam em sistemas operacionais e realizam operações em banco de dados instalados no sistema.

UNIDADE 2 – ARQUIVOS DE TEXTO, DOCUMENTOS, PLANILHAS E APRESENTAÇÕES

MÓDULO 3 – CNAB – CENTRO NACIONAL DE AUTOMAÇÃO BANCÁRIA

01

1 - FEBRABAN – FEDERAÇÃO BRASILEIRA DE BANCOS

Olá, seja bem-vindo a mais um módulo de estudo. Nas unidades passadas vimos conceitos básicos e formas de manipular arquivos baseados em texto, como um programa de computador pode manipular esses arquivos, e vimos também alguns tipos de arquivos como INI, LOG, XML e scripts.

Neste módulo, iremos aprender como as instituições bancárias utilizam um arquivo **de texto plano**, em um formato especial chamado **CNAB (Centro Nacional de Automação Bancária)**, para trocar informações bancárias (como compensação de cheques, pagamento de contas etc.).

Se você um dia for trabalhar como programador de uma instituição bancária ou for programar um sistema que realiza operações bancárias, provavelmente seu sistema precisará trocar informações no formato CNAB.

Atenção: para você mesmo fazer testes e verificar os exemplos citados neste módulo, utilizaremos um *software* livre para edição de arquivos. Como sugestão, baixe e instale o “Notepadd++”, disponível gratuitamente neste endereço: http://notepad-plus-plus.org/.

02

A FEBRABAN - Federação Brasileira de Bancos é a principal entidade representativa do setor bancário brasileiro. Fundada em 1967, na cidade de São Paulo, é uma associação sem fins lucrativos que tem o compromisso de fortalecer o sistema financeiro e suas relações com a sociedade e contribuir para o desenvolvimento econômico, social e sustentável do País.

O objetivo da Federação é representar seus associados em todas as esferas do governo – poderes Executivo, Legislativo e Judiciário e entidades representativas da sociedade, para o aperfeiçoamento do sistema normativo, a melhoria continuada dos serviços e a redução dos níveis de risco. Também busca concentrar esforços que favoreçam o crescente acesso da população aos produtos e serviços financeiros.

O quadro associativo da entidade conta com 121 instituições financeiras associadas, que representam 93% do patrimônio líquido e 97% dos ativos totais do sistema bancário brasileiro.

03

2 - CNAB 240

Quando estudamos sobre o SIGAD, vimos que a organização CONARQ regulamenta e mantém o e-ARQ, que é um padrão para *softwares* SIGAD no Brasil. De forma análoga, a FEBRABAN regulamenta e mantém o padrão CNAB, cuja versão foco do nosso estudo é o CNAB 240.

O CNAB é um modelo padrão para instituições trocarem informações financeiras.

O CNAB é amplamente utilizado por bancos e instituições financeiras. Dentre as operações mais comuns, podemos citar:

- Para uma empresa qualquer enviar dados de boletos de pagamento a serem pagos para um banco.
- Para uma empresa qualquer enviar dados de pagamento de salário para um banco realizar os depósitos e transferências.
- Para dois bancos trocarem informações sobre compensação de cheques.
- Para dois bancos trocarem informações sobre pagamentos.
- Para uma empresa enviar dados de pagamentos feitos em cartão de crédito para o banco.

E outras tantas possibilidades.

Cada tipo de operação tem um formato e um *layout* específico dentro do padrão CNAB. Portanto, o*layout* para pagamento de boleto bancário é diferente do *layout* para pagamento de salários.

04

Vamos ver um exemplo prático para ficar mais claro. Suponha que uma empresa tenha 200 funcionários. Seria muito difícil para a empresa todo início de mês realizar o pagamento de todos os 200

funcionários em um único dia. Para isso a empresa poderia fazer um acordo com um Banco e criar um sistema informatizado que todo dia primeiro do mês a empresa envia um arquivo com o pagamento a ser feito para o banco.

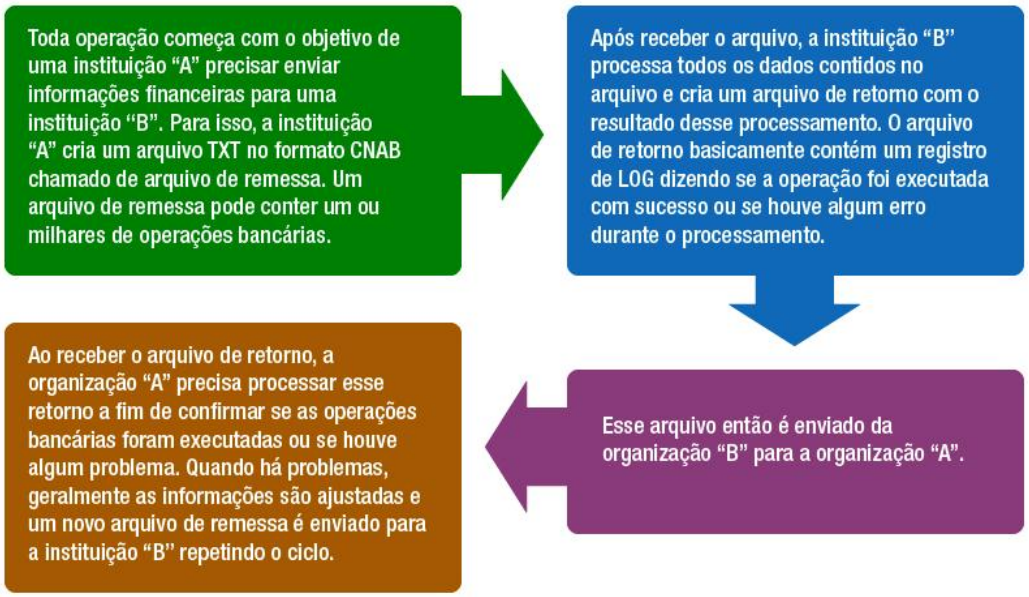
Ao receber o arquivo, o banco identifica a conta corrente da empresa e a conta corrente de cada funcionário, bem como o valor do salário e cada um. Em seguida, o banco faz uma transferência do valor de cada salário da conta corrente da empresa para a conta corrente de cada um dos funcionários. Ao término do processamento dos 200 pagamentos, o banco gera um arquivo de retorno com o resultado dos pagamentos feitos e envia esse arquivo de volta para a empresa.

O CNAB pode ser gratuitamente obtido a partir deste link: http://goo.gl/toj9Dr.

05

2.1. Como funciona o processamento básico do padrão definido pela FEBRABAN?

O CNAB prevê sempre um par de ações, denominadas **remessa** e **retorno**. O funcionamento padrão de troca de dados entre instituições é o seguinte:



06

2.2. Formato e *layout* do arquivo compatível com o modelo CNAB?

Lembra quando você estudou arquivo CSV? Lembra que as vírgulas separavam as informações? De forma diferente, os arquivos no formato CNAB possuem formato texto com colunas fixas definidas. Cada linha pode ter um significado diferente e as informações estão posicionadas em colunas específicas.

Exemplo de informações de pagamento de salários dos funcionários de uma empresa no formato CSV:

```
1  Nome,CPF,Banco,Agencia,ContaCorrente,Salario
2  Andre Passos,45598754211,001,2548,25487,3200.00
3  Gustavo Mendonca,9875465452,341,5874,458778,4250.00
4  Paulo Motta, 54875436954,001,5478,558458,5230.00
```

Exemplo de informações de pagamento de salários dos funcionários de uma empresa no formato CNAB:

```
1  02Andre Passos          45598754211001254825487300200000
2  02Gustavo Mendonca      98754654523415874458778400250000
3  02Paulo Motta           54875436954001547855845800523000
```

Observe no exemplo acima que no formato CVS é mais fácil encontrar a informação completa, enquanto que no formato CNAB letras, números e espaço estão todos juntos. Se não soubermos onde inicia e onde termina uma informação fica impossível conseguir ler cada uma das informações da cada pessoa.

07

Então, por que utilizar o formato CNAB já que ele parece ser tão confuso? Elencamos alguns motivos:

a. O **CNAB permite você incluir diversos tipos de informação no mesmo arquivo**, enquanto que no CSV só é possível mandar um tipo de informação em cada arquivo. No exemplo que demos da empresa enviando dados de pagamentos de funcionários, se utilizássemos o formato CSV, seria necessário um arquivo com os dados da empresa que efetuará os pagamentos e outro arquivo com os dados dos empregados. Se fosse necessário realizar várias operações bancárias, seriam necessários muitos arquivos CSV, um para cada tipo de dados. Já no formato CNAB, um único arquivo contém todas as informações necessárias.
b. O **CNAB 240, por ser arquivo de texto plano, pode ser lido por tecnologias muito antigas**, como computadores de grande porte, programas em Cobol e outros. Os computadores e programas feitos na década de 70 são aptos a ler informações tabuladas como o arquivo padrão CNAB, mas tem muita dificuldade em interpretar arquivos CSV. Acredite, há bancos que utilizam essas tecnologias ainda nos dias atuais pelo simples fato de estarem funcionando bem e, além do alto custo para mudar para uma tecnologia nova, não haveria nenhum benefício em trocar de tecnologia.

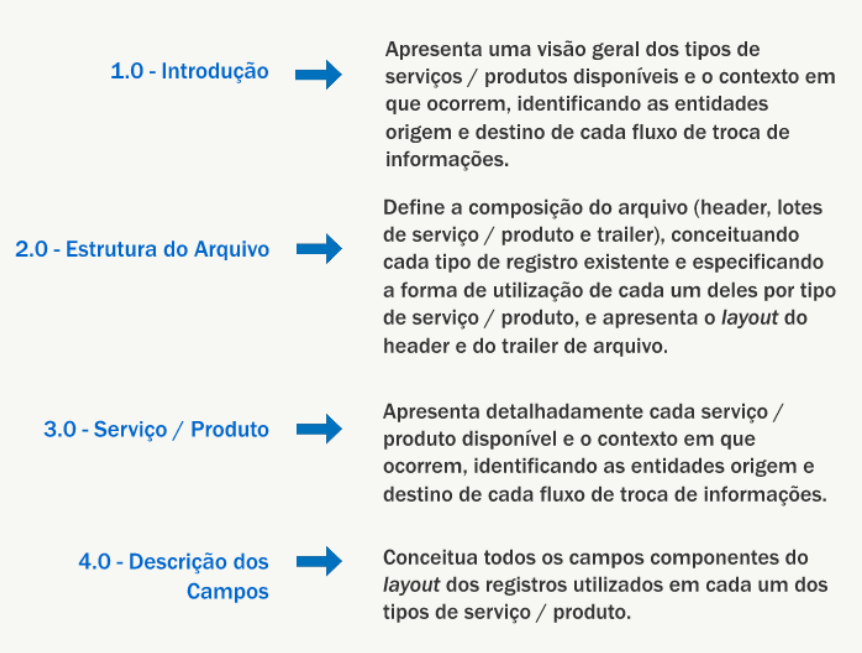
Então, por que não utilizar o formato XML, já que ele permite que vários tipos de dados sejam armazenados no mesmo arquivo? Pelo mesmo motivo citado na letra “b” acima, o uso de tecnologias antigas não são compatíveis com o formato XML. Se XML fosse um padrão bancário, todos os bancos e

empresas que utilizam tecnologias antigas precisariam atualizar seus sistemas. E isso seria caríssimo e impraticável!

08

2.3. Como o documento do CNAB 240 está estruturado?

O documento do CNAB 240 está dividido nos seguintes tópicos:



09

3 - Estudando o CNAB 240

Estudaremos o CNAB 240 em duas etapas: primeiro, iremos entender como um arquivo no padrão CNAB 240 está constituído, que tipo de informação está presente em cada linha e quais campos/informações cada linha contém. Essa informação está descrita no documento CNAB-240. Depois, iremos entender como um *software* consegue ler e interpretar um arquivo CNAB, bem como de que forma um *software* pode criar um arquivo CNAB 240. Para isso usaremos fragmentos de código fonte de programas para estudar e entender como fazer. Futuramente, de posse dessas duas informações, quando você estudar uma linguagem de programação, você facilmente conseguirá criar um sistema que lê, grava e interpreta informações de arquivos no formato CNAB 240.

Como já foi falado, um arquivo no formato CNAB 240, antes de tudo é um arquivo de texto puro. Portanto, qualquer editor de texto pode ser utilizado para analisar o conteúdo de um arquivo CNAB ou até mesmo criar um arquivo no formato CNAB. Obviamente, não faz sentido para uma empresa alguém criar ou ler manualmente arquivos no formato CNAB. É necessário um sistema que faça isso.

Um arquivo no padrão CNAB 240 possui várias linhas chamadas de **registros**. Cada registro representa um tipo de informação. Há sete tipos de registros.

A primeira linha do arquivo é chamada de **header** e representa o cabeçalho, a última linha do arquivo é chamada de **trailer** e representa o rodapé do arquivo. As linhas intermediárias representam o conteúdo do arquivo propriamente dito.

Cada tipo de conteúdo fica organizado em **lotes** diferentes. Ou seja, se o arquivo contém informação de pagamentos de boletos e informação de transferências bancárias, então esse arquivo tem dois lotes, um para cada tipo de informação.

10

Cada lote está organizado da seguinte maneira:

- inicia com uma linha que indica o início do lote (**header do lote**),
- depois vêm os registros que informam as operações bancárias (chamadas de **registros detalhe**),
- termina com uma linha que indica o final do lote (chamada de **trailer do lote**).

Cada tipo de linha possui um identificador de linha na oitava posição. Ou seja, o oitavo caractere de cada linha define o tipo de linha que estamos tratando. São sete os tipos de linhas, conforme descritos a seguir.

Quando o oitavo caractere é igual a...	Significa que esse identificador de linha...
0	Indica que é o HEADER do arquivo.
1	Indica que é o HEADER do lote.
2	Indica informações iniciais do lote (opcional).
3	Indica cada um dos registros detalhe (operações bancárias)
4	Indica informações finais do lote (opcional).
5	Indica o trailer do lote.
9	Indica o trailer do arquivo.

Identificadores de linha e seus significados.

11

Em um arquivo CNAB:

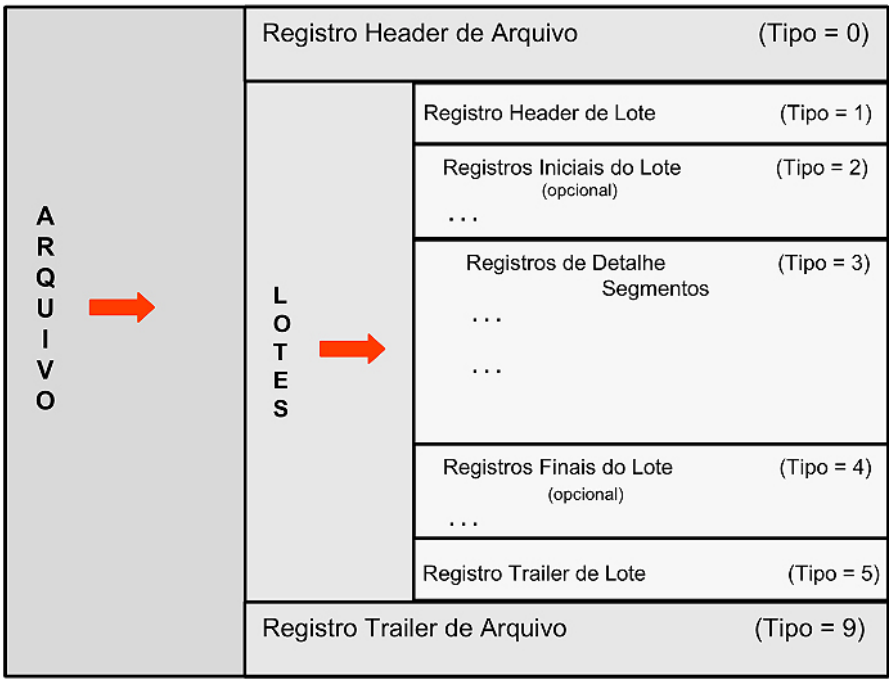
- Só podemos ter uma linha (um registro) do tipo 0 (header de arquivo) que é a primeira linha do arquivo. Somente a primeira linha pode ser do tipo 0. Sempre a primeira linha tem que ser do tipo 0. Se um arquivo contiver outra linha com o identificador 0, então o arquivo é considerado inválido (erro de *layout* de arquivo).
- Só podemos ter uma linha do tipo 9 (trailer de arquivo) que é a última linha do arquivo.

Somente a última linha pode ser do tipo 9. Sempre a última linha tem que ser do tipo 9. Se um arquivo contiver outra linha com identificador 9, então o arquivo é considerado inválido (erro de*layout* de arquivo).

- Cada lote (tipo de registro) deve possuir, obrigatoriamente uma linha com identificador 1, depois várias linhas que com identificadores 2 ou 3, depois várias linhas com identificador 3, e depois uma linha com identificador 4 ou 5. A primeira linha do lote sempre possuirá identificador 1 e a última linha do lote sempre possuirá identificador 5. Se o lote começar com linhas cujo identificador seja 2, 3, 4 ou 5, então o arquivo será considerado inválido. Cada lote só pode conter um único tipo de serviço bancário.
- Dentro de um lote, só pode haver uma linha tipo 1, uma linha tipo 2, uma linha tipo 4 e uma linha tipo 5, mas dentro do lote pode haver uma ou até milhares de linhas tipo 3.
- Caso o arquivo contenha mais de um tipo de operação, haverá um conjunto de linhas de lote para cada um dos tipos de operação presente.
- Somente as linhas do tipo 3 contém os dados das operações bancárias, as demais linhas são usadas apenas para controle, validação e identificação da empresa que emitiu o arquivo CNAB.

12

Dessa forma, explicados os requisitos iniciais da estrutura do arquivo CNAB 240, temos esse diagrama que ilustra o que foi citado neste capítulo:



Layout padrão de um arquivo CNAB 240.

Exemplo prático: Suponha que um arquivo hipotético tenha 1 lote com 3 registros de depósito bancário e 1 lote com 2 registros de pagamento de boletos. Dessa forma, o mínimo esperado é que esse arquivo possua 11 linhas, onde cada linha do arquivo representa:

Sequência da Linha	Identificador da Linha	Significado
1	0	Reader do arquivo – início do arquivo.
2	1	Reader do primeiro lote de depósitos bancários
3	3	Linha detalhe – primeiro depósito a ser feito
4	3	Linha detalhe – segundo depósito a ser feito
5	3	Linha detalhe – terceiro depósito a ser feito
6	5	Trailer de lote – fim dos depósitos bancários
7	1	Reader do primeiro lote de pagamento de boletos
8	3	Linha detalhe – primeiro depósito a ser feito
9	3	Linha detalhe – segundo depósito a ser feito
10	5	Trailer de lote – fim dos pagamento de boletos
11	9	Trailer do arquivo – fim do arquivo

13

3.1. Tipos de lote (ou tipos de arquivo detalhe)

O CNAB 240 prevê **14 tipos de operações**. Cada tipo de operação possui um identificador único. Estas são as operações previstas no CNAB:

a. Pagamento através de Crédito em Conta Corrente, Cheque, OP, DOC ou Pagamento com Autenticação;
b. Débito em Conta Corrente;
c. Extrato de Conta Corrente para Conciliação Bancária;
d. Pagamento de Títulos de Cobrança;
e. Bloqueto Eletrônico (Captura de Títulos em Cobrança);
f. Títulos em Cobrança;
g. Alegação do Sacado;
h. Vendor;
i. Custódia de Cheques;
j. Extrato para Gestão de Caixa;
k. Empréstimo por Consignação;
l. Pagamento de Tributos (com ou sem código de barras);
m. Consulta de Tributos a Pagar.
n. Compror (financiamento).

14

3.2. Que caracteres são aceitos no formato CNAB 240?

O nome CNAB 240 significa que cada linha tem exatamente 240 caracteres. É nesse espaço de 240 caracteres que todas as informações de cada tipo de linhas têm disponíveis. Cada linha não pode ter nem mais nem menos que 240 caracteres, mas exatos 240.

Na maioria dos tipos de informações você verá que há espaços que não são utilizados. Esses espaços devem ficar em branco.

Em termos de caracteres, basicamente somente são aceitas as letras (sempre em maiúsculo), números e os demais símbolos do teclado (como o @, #, &, % etc.), letras acentuadas ou cedilha não são aceitos. Os números não aceitam pontuação, marcação de decimal, milhar ou símbolo monetário.

Dessa forma, as informações precisam ser adaptadas para serem representadas corretamente. Essas são adaptações que precisamos fazer:

- Campos do tipo data: precisamos excluir as barras; dias e meses menores que 10 precisam ter o caractere zero à esquerda. Dessa forma, a data 1/3/2014 é representada como 01032014 no arquivo. Quando nosso programa ler esse tipo de informação, precisaremos adicionar as barras para formatar corretamente o campo como data.
- Campos do tipo decimal: campos decimais (como valores, moedas) não podem conter vírgula ou ponto, ainda, precisam conter zeros a esquerda conforme a largura do campo. Imagine um campo com 12 posições, para representar o valor de R$ 1.234,56 nesse formato, teremos: 000000123456. Quando nosso programa ler esse tipo de informação, precisaremos dividir o valor numérico por 100 para calcularmos corretamente o valor numérico original.
- Campos do tipo texto: sempre com letras em maiúsculas, sem letras acentuadas ou cedilha, com espaços em branco à direita. Portanto, um campo com 20 posições para o nome da cidade, sendo a cidade de nome “Brasília”, ficaria assim armazenada: “BRASILIA “.

15

4 - *LAYOUT* DO ARQUIVO (SETE TIPOS DE LINHA)

Nesta seção vamos ver um pouco sobre o *layout* dos sete tipos de linha. O manual do CNAB contém a informação completa de todos os tipos, portanto, consulte o manual do CNAB 240 para saber em detalhes todos os tipos de linha.

4.1. Registro do tipo “Header de Arquivo”

Conforme já falamos, a linha de **header de arquivo** é a primeira linha do arquivo.

Na linha de **header de arquivo** estão presentes informações sobre o banco de origem ou de destino do arquivo, a empresa que emitiu o arquivo e informações sobre o arquivo.

Lembre-se, temos 240 caracteres para conter todas as informações citadas. Você não precisa decorar nenhum *layout*, apenas entender como as informações são descritas no arquivo.

Como temos uma limitação na largura do texto deste documento, a título de aprendizado, a seguir iremos apresentar apenas parte do conteúdo do *layout* (os primeiros 52 dos 240 caracteres do *layout*).

O conteúdo completo você encontra no manual do CNAB 240.

16

Nos primeiros 52 dos 240 caracteres temos o seguinte *layout*:

- Da posição 1 (primeiro caractere da linha) até a posição 3 (terceiro caractere da linha), refere-se ao código do banco que está recebendo ou emitindo o arquivo. Exemplo: Banco Itaú = 341, Banco do Brasil = 001.
- Da posição 4 até a posição 7 (total de 4 caracteres), temos o lote do serviço, que sempre será "0000" no caso do arquivo de header de arquivo.
- Na posição 8 (apenas um caractere) está definida o tipo de linha, no caso, linha de header, código 0. É a análise deste caractere na oitava posição que diz que esta é uma linha de header de arquivo.
- Da posição 9 à posição 17 devem haver nove espaços em branco.
- Na posição 18 deve constar o tipo de inscrição de empresa: 0 para isento de inscrição ou 2 para CNPJ.
- Da posição 19 até a posição 32 (total de 34 caracteres) deve vir o CNPJ da empresa ou somente zeros se ela for isenta.
- Da posição 33 até a posição 52 deve vir o código de convênio do banco, que é o código adotado pelo Banco para identificar o Contrato entre este e a Empresa Cliente.

Dessa forma, um trecho hipotético de linha de header seria:

```
34100000         00000000000000000000225254254000125000000000000000248291...
    341 – indica que se trata do banco Itaú;
        0000 – valor zero para lote de header;
            0 – identificador de arquivo de header;
                    - nove espaços em branco;
                    2 – indica que a identificação da empresa será por meio de CNPJ;
                                    225254254000125 – o CNPJ da empresa;
                                                      248291 – id da empresa no banco
```

17

Um resumo do *layout* completo do registro header de arquivo você vê na tabela abaixo:

CAMPO					POSIÇÃO DE	POSIÇÃO ATÉ	Nº DIG
01.0	Controle	Banco		Código do Banco na Compensação	1	3	3
02.0		Lote		Lote de Serviço	4	7	4
03.0		Registro		Tipo de Registro	8	8	1
04.0	CNAB			Uso Exclusivo FEBRABAN / CNAB	9	17	9
05.0	Empresa	Inscri-ção	Tipo	Tipo de Inscrição da Empresa	18	18	1
06.0			Número	Número de Inscrição na Empresa	19	32	14
07.0		Convênio		Código do Convênio no Banco	33	52	20
08.0		Conta Cor-rente	Agência	Código	Agência Mantenedora da Conta	53	57	5
09.0				DV	Dígito Verificador da Agência	58	58	1
10.0			Conta	Número	Número da Conta Corrente	59	59	12
11.0				DV	Dígito Verificador da Conta	71	71	1
12.0			DV		Dígito Verificador da Ag/Conta	72	72	1
13.0		Nome			Nome da Empresa	73	73	30
14.0	Nome do Banco				Nome do Banco	103	103	30
15.0	CNAB				Uso Exclusivo FEBRABAN / CNAB	133	133	10
16.0	Arquivos	Código			Código Remessa / Retorno	143	143	1
17.0		Data de Geração			Data de Geração do Arquivo	144	144	8
18.0		Hora de Geração			Hora de Geração do Arquivo	152	152	6
19.0		Sequência (NSA)			Número Sequencial do Arquivo	158	158	6
20.0		Layout do Arquivo			Nº da Versão do Layout do Arquivo	164	164	3
21.0		Densidade			Densidade de Gravação do Arquivo	167	167	5
22.0	Reservado Banco				Para Uso Reservador do Banco	172	172	20
23.0	Reservado Empresa				Para Uso Reservado da Empresa	192	192	20
24.0	CNAB				Uso Exclusivo FEBRABAN / CNAB	212	240	29

***Layout* da linha de header de arquivo.**

Observe que há basicamente 3 blocos de informação na linhas de header e um espaço não utilizado:

- Da posição 1 à posição 17, da posição 103 à posição 132 e da posição 172 à posição 191 contém dados do banco;
- Da posição 18 à posição 102 e da posição 192 à posição 211 temos dados da empresa que emitiu ou receberá o arquivo;
- Da posição 143 à posição 171 temos dados sobre a criação do arquivo;
- Da posição 212 à posição 240 temos um espaço não utilizado.

18

4.2. Registro do tipo “Trailer de Arquivo”

Conforme também já falamos, a linha de **trailer de arquivo** é a última linha do arquivo.

Na linha **trailer de arquivo** estão presentes informações de controle do arquivo e totais. Essas informações são utilizadas para validar se nenhuma linha do arquivo foi inserida, alterada ou excluída durante a transmissão do arquivo.

É uma informação que o programa que processa essa linha precisa realizar testes validando se as informações dessa linha conferem com o processamento proposto. Caso ocorra alguma informação incoerente, o processamento de todo o arquivo deve ser cancelado, e um arquivo de retorno informando erro de conteúdo deve ser criado. Ou seja, ou processamos todas as informações do arquivo ou o arquivo inteiro deve ter seu processamento cancelado, não podemos processar apenas uma parte das operações bancárias.'

O *layout* do header é o mais simples de todos os tipos de linhas, mesmo assim, ocupa 240 caracteres da seguinte forma:

- Da posição 1 (primeiro caractere da linha) até a posição 3 (terceiro caractere da linha), repete o mesmo código do banco citado na linha de Header de arquivo.
- Da posição 4 até a posição 7 (total de 4 caracteres), temos o lote do serviço, que sempre será "9999" no caso do arquivo de trailer de arquivo.
- Na posição 8 (apenas um caractere) está definida o tipo de linha, no caso, linha de trailer de arquivo, código 9. É a análise deste caractere na oitava posição que diz que esta é uma linha de trailer de arquivo.
- Da posição 9 à posição 17 devem haver nove espaços em branco.
- Da posição 18 até a posição 23 (total de 6 caracteres) deve vir a quantidade de lotes (tipos de registro) do arquivo, que deve ser no mínimo 000001 e no máximo 000014 (pois só existem 14 tipos de registro, conforme explicamos no subitem 6.1 deste módulo).
- Da posição 24 até a posição 29 (total de 6 caracteres) deve vir a quantidade de linhas do arquivo, que deve ser no mínimo 000005 (um reader de arquivo, um reader de lote, um arquivo detalhe, um trailer de lote e um trailer de arquivo) e no máximo 999999.
- Da posição 30 até a posição 35 (total de 6 caracteres) deve vir a quantidade de lotes de Conciliação Bancária enviados no arquivo. Somatória dos registros de tipo 1 e Tipo de Operação = 'E'.
- Da posição 36 à posição 240 deve haver 205 espaços em branco.

19

Um trecho hipotético de linha de header seria:

```
34199999         000003000037000000                          ... (até o 240º caractere)
    341 – indica que trata-se do banco Itaú;
       9999 – valor 9999 para lote de trailer de arquivo;
         9 – identificador de arquivo de header;
                 - nove espaços em branco;
                 000003 – indica que há 3 lotes neste arquivo;
                     000037 – indica que há 37 registros detalhe neste arquivo;
                         000000 – indica que não há nenhum lote de conciliação neste
arquivo;
                                - 205 espaços em branco.
```

Um resumo do *layout* completo do registro trailer de arquivo você vê na tabela abaixo:

CAMPO				POSIÇÃO		Nº DIG
				DE	ATÉ	
01.9	Controle	Banco	Código do Banco na Compensação	1	3	3
02.9		Lote	Lote de Serviço	4	7	4
03.9		Registro	Tipo de Registro	8	8	1
04.9	CNAB		Uso Exclusivo FEBRABAN / CNAB	9	17	9
05.9	Controle	Qtde. de Lotes	Quantidade de Lotes do Arquivo	18	23	6
06.9		Qtde. de Registros	Quantidade de Registros do Arquivo	24	29	6
07.9		Qtde. de Contas Concil.	Qtde de Contas p/ Conc. (Lotes)	30	35	6
08.9	CNAB		Uso Exclusivo FEBRABAN / CNAB	36	240	205

***Layout* da linha de trailer de arquivo.**

20

4.3. Registro do tipo “header de lote”

A linha de header de lote é a linha que identifica um tipo de serviço bancário. Abaixo dela aparecerão uma ou mais operações bancárias do mesmo tipo.

Após a última operação bancária do mesmo tipo deverá aparecer uma linha de trailer de lote. Em um arquivo CNAQ 240 pode haver até 14 lotes, representando os 14 tipos diferentes de serviços bancários possíveis.

Nesta linha estão presentes informações sobre:

- Controle do arquivo: informações sobre o banco (o mesmo da linha de header de arquivo), o tipo de lote (que para este tipo de linha deve ser um número sequencial que comece em 0001 e vá aumentando de um a um para cada novo lote presente no arquivo (máximo de 14 lotes), o identificador da linha (que para header de lote sempre será igual a 1);
- Dados da empresa: como CGC, cadastro no banco, dados conta corrente e nome da empresa;
- Endereço da empresa: rua, número, cidade, uf, etc.
- Indicativo da forma de pagamento: 01 para débito em conta corrente, 02 para financiamento ou empréstimo e 03 para cartão de débito.
- Ainda, quando se tratar de arquivo de retorno, um status sobre o tipo de retorno (normal, erro de registro, erro de linha, falta de fundos etc.).

Um resumo do *layout* completo do registro reader de lote você vê na tabela abaixo:

CAMPO						POSIÇÃO DE	POSIÇÃO ATÉ	Nº DIG
01.1	Controle	Banco			Código do Banco na Compensação	1	3	3
02.1		Lote			Lote de Serviço	4	7	4
03.1		Registro			Tipo de Registro	8	8	1
04.1	Serviço	Operação			Tipo de Operação	9	9	1
05.1		Serviço			Tipo de Serviço	10	11	2
06.1		Forma Lançamento			Forma de Lançamento	12	13	2
07.1		Layout do Lote			Nº da Versão do Layot do Lote	14	16	3
08.1	CNAB				Uso Exclusivo FEBRABAN / CNAB	17	17	1
09.1	Empresa	Inscrição	Tipo		Tipo de Inscrição da Empresa	18	18	1
10.1			Número		Número de Inscrição da Empresa	19	32	14
11.1		Convênio			Código do Convênio no Banco	33	52	20
12.1		Conta Corrente	Agência	Código	Agência Mantenedora da Conta	53	57	5
13.1				DV	Dígito Verificador da Agência	58	58	1
14.1			Conta	Número	Número da Conta Corrente	59	70	12
15.1				DV	Dígito Verificador da Conta	71	71	1
16.1			DV		Dígito Verificador da Ag/Conta	72	72	1
17.1		Nome			Nome da Empresa	73	102	30
18.1	Informação 1				Mensagem	103	142	40
19.1	Endereço da Empresa	Logradouro			Nome da Rua, Av. Pça, Etc	143	172	30
20.1		Número			Número do Local	173	177	5
21.1		Complemento			Casam Apto, Sala, Etc	178	192	15
22.1		Cidade			Nome da Cidade	193	212	20
23.1		CEP			CEP	213	217	5
24.1		Complemento CEP			Complemento do CEP	218	220	3
25.1		Estado			Sigla do Estado	221	222	2
26.1	Indicativo de Forma de Pagamento				Indic. da Forma de Pgto do Serviço	223	224	2
27.1	CNAB				Uso Exclusivo FEBRABAN / CNAB	225	230	6
28.1	Ocorrências				Código das Ocorrências p/ Retorno	231	240	10

***Layout* da linha de header de lote.**

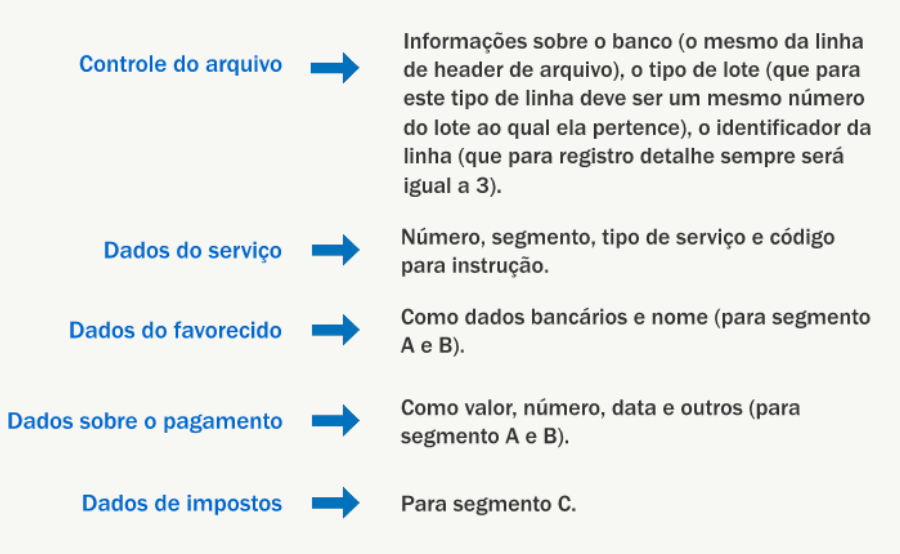
21

4.4. Registro do tipo “registro detalhe”

A linha de registro de detalhe é onde estão as operações bancárias a serem feitas. Cada tipo de operação é agrupado em lotes.

Há cerca de 20 *layouts* diferentes para poder representar os 14 tipos de operações bancárias possíveis. Cada *layout* é chamado de segmento, os segmentos vão do “A” ao “Z”, mas não possuem todas as possibilidades de letras (por exemplo, não existe segmento “D”). Abaixo de uma linha registro detalhe pode aparecer nenhuma, uma ou mais linhas de registro de detalhe. Após a última linha de registro detalhe obrigatoriamente aparecerá uma linha de trailer de lote.

Nesta linha estão presentes informações sobre:



22

Um resumo do *layout* completo do registro detalhe do segmento “A” você vê na tabela abaixo:

CAMPO						POSIÇÃO		Nº DIG	Nº DEC	FORMATO	DEFAULT
						DE	ATÉ				
01.3A	Controle	Banco			Código do Banco na Compensação	1	3	3	-	Num	
02.3A		Lote			Lote de Serviço	4	7	4	-	Num	
03.3A		Registro			Tipo de Registro	8	8	1	-	Num	'3'
04.3A	Serviço	Nº do Registro			Nº Sequêncial do Registro no Lote	9	13	5	-	Num	
05.3A		Segmento			Cód. de Segmento do Reg. Detalhe	14	14	1	-	Alfa	'A'
06.3A		Movimento	Tipo		Tipo de Movimento	15	15	1	-	Num	
07.3A			Código		Código da Instrução p/ Movimento	16	17	2	-	Num	
08.3A	Favorecido	Câmara			Código da Câmara Centralizada	18	20	3	-	Num	
09.3A		Banco			Código do Banco do Favorecido	21	23	3	-	Num	
10.3A		Conta Corrente	Agência	Código	Ag. Mantenedora da Cta do Favor.	24	28	5	-	Num	
11.3A				DV	Digito Verificador da Agência	29	29	1	-	Alfa	
12.3A			Conta	Número	Número da Conta Corrente	30	41	12	-	Num	
13.3A				DV	Dígito Verificador da Conta	42	42	1	-	Alfa	
14.3A			DV		Dígito Verificador da Ag/Conta	43	43	1	-	Alfa	
15.3A		Nome			Nome do Favorecido	44	73	30	-	Alfa	
16.3A	Crédito	Seu Número			Nº do Docum. Atribuído p/ Empresa	74	93	20	-	Alfa	
17.3A		Data Pagamento			Data do Pagamento	94	101	8	-	Num	
18.3A		Moeda	Tipo		Tipo de Moeda	102	104	3	-	Alfa	
19.3A			Quantidade		Quantidade da Moeda	105	119	10	5	Num	
20.3A		Valor Pagamento			Valor do Pagamento	120	134	13	2	Num	
21.3A		Nosso Número			Nº do Docum. Atribuído pelo Banco	135	154	20	-	Alfa	
22.3A		Data Real			Data Real da Efetivação do Pagto	155	162	8	-	Num	
23.3A		Valor Real			Valor Real da Efetivação do Pagto	163	177	13	2	Num	
24.3A	Informação 2				Outras Informações - Vide Formatação em G031 para Identificação de Depósito Judicial e Pagto. Salários de servidores pelo SIAPE	178	217	40	-	Alfa	
25.3A	Código Finalidade Doc				Compl. Tipo Serviço	218	219	2	-	Alfa	
26.3A	Código Finalidade TED				Código finalidade da TED	220	224	5	-	Alfa	
27.3A	Código Finalidade Complementar				Complemento de finalidade pagto.	225	226	2	-	Alfa	
28.3A	CNAB				Uso Exclusivo FEBRABAN / CNAB	227	229	3	-	Alfa	Brancos
29.3A	Aviso				Aviso ao Favorecido	230	230	1	-	Num	
29.3A	Ocorrências				Código das Ocorrências p/ Retorno	231	240	10	-	Alfa	

Clique aqui para ver um resumo do *layout* completo do registro detalhe do segmento "J".

Clique aqui para ver um resumo do *layout* completo do registro detalhe do segmento "N".

***Layout* da linha de detalhe tipo "J"**

CAMPO				POSIÇÃO DE	POSIÇÃO ATÉ	Nº DIG	Nº DEC	FORMATO	DEFAULT
01.3J	Controle	Banco	Código do Banco na Compensação	1	3	3	-	Num	
02.3J		Lote	Lote de Serviço	4	7	4	-	Num	
03.3J		Registro	Tipo de Registro	8	8	1	-	Num	'3'
04.3J	Serviço	Nº do Registro	Nº Sequêncial do Registro no Lote	9	13	5	-	Num	
05.3J		Segmento	Cód. de Segmento do Reg. Detalhe	14	14	1	-	Alfa	'J'
06.3J		Movimento – Tipo	Tipo de Movimento	15	15	1	-	Num	
07.3J		Movimento – Código	Código da Instrução p/ Movimento	16	17	2	-	Num	
08.3J	Pagamento	Código Barras	Código Barras	18	61	44	-	Num	
09.3J		Nome de Cedente	Nome de Cedente	62	91	30	-	Alfa	
10.3J		Data Vencimento	Data Vencimento (Nominal)	92	99	8	-	Num	
11.3J		Valor do Título	Valor do Título (Nominal)	100	114	13	2	Num	
12.3J		Desconto	Valor do Desconto + Abatimento	115	129	13	2	Num	
13.3J		Acréscimos	Valor da Mora + Multa	130	144	13	2	Num	
14.3J		Data Pagamento	Data do Pagamento	145	152	8	-	Num	
15.3J		Valor Pagmento	Valor do Pagmento	153	167	13	2	Num	
16.3J		Quantidade Moeda	Quantidade da Moeda	168	182	10	5	Num	
17.3J		Referência Sacado	Nº do Docto Atribuído pela Empresa	183	202	20	-	Alfa	
18.3J	Nosso Número		Nº do Docto Atribuído pelo Banco	203	222	20	-	Alfa	
19.3J	Código de Moeda		Código da Moeda	223	224	2	-	Num	
20.3J	CNAB		Uso Exclusivo FEBRABAN / CNAB	225	230	6	-	Alfa	Brancos
21.3J	Ocorrências		Código das Ocorrências p/ Retorno	231	240	10	-	Alfa	

***Layout* da linha de detalhe tipo "N"**

CAMPO				POSIÇÃO DE	POSIÇÃO ATÉ	Nº DIG	Nº DEC	FORMATO	DEFAULT
01.3N	Controle	Banco	Código do Banco na Compensação	1	3	3	-	Num	
02.3N		Lote	Lote de Serviço	4	7	4	-	Num	
03.3N		Registro	Registro de Lote	8	8	1	-	Num	'3'
04.3N	Serviço	Nº do Registro	Nº Sequêncial do Registro no Lote	9	13	5	-	Num	
05.3N		Segmento	Cód. de Segmento do Reg. Detalhe	14	14	1	-	Alfa	'N'
06.3N		Movimento – Tipo	Tipo de Movimento	15	15	1	-	Num	
07.3N		Movimento – Código	Código da Instrução de Movimento	16	17	2	-	Num	
08.3N	Pagamento	Seu Número	Nº do Docto Atribuído pela Empresa	18	37	20	-	Alfa	
09.3N		Nosso Número	Nº do Docto Atribuído pelo Banco	38	57	20	-	Alfa	
10.3N		Contribuinte	Nome do Contribuinte	58	87	30	-	Alfa	
11.3N		Data Pagamento	Data do Pagamento	88	95	8	-	Num	
12.3N		Valor Pagamento	Valor Total do Pagmento	96	110	13	2	Num	
13.3N	Informações Complementares		Informações Complementares de acordo com o respectivo tributo	111	230	120	-	Alfa	
14.3N	Ocorrências		Código das Ocorrências p/ Retorno	231	240	10	-	Alfa	G059

23

4.4.1. Outros *layouts* para registro detalhe

O CNAB oferece ainda outros detalhes específicos para pagamento de GPS, IPVA, DARF, DPVAT, Licenciamento de veículos, FGTS e outros tipos de pagamento. Consulte o manual do CNAQ 240 para detalhes.

O quadro abaixo apresenta um resumo de todos os *layouts* que existem:

LOTE	SERVIÇO / PRODUTO	SEGMENTOS	
		REMESSA	RETORNO
Pagamento através de Crédito em Conta Corrente, Cheque, OP, DOC ou Pagamento com Autenticação	Pagamentos	A (Obrigatório) B (Opcional) C (Opcional)	A (Obrigatório) B (Opcional) C (Opcional)
Débito em Conta Corrente	Débito em Conta Corrente	A (Obrigatório) B (Opcional) C (Opcional)	A (Obrigatório) B (Opcional) C (Opcional)
Extrato de Conta Corrente para Conciliação Bancária	Extrato de Conta Corrente para Conciliação Bancária		E (Obrigatório)
Pagamento de Títulos de Cobrança	Pagamentos	J (Obrigatório)	J (Obrigatório)
Bloqueto Eletrônico (Captura de Títulos de Cobrança)	Cobrança		G (Obrigatório) H (Opcional) Y (Opcional)
Títulos de Cobrança	Cobrança	P (Obrigatório) Q (Obrigatório) R (Opcional) S (Opcional) Y (Opcional)	T (Obrigatório) U (Obrigatório) Y (Opcional)
Alegação do Sacado	Cobrança	Y (Obrigatório)	Y (Obrigatório)
Vendor	Vendor	K (Obrigatório) L (Obrigatório)	K (Obrigatório) M (Obrigatório) N (Obrigatório)
Custódia de Cheques	Custódia de Cheques	D (Obrigatório)	D (Obrigatório)
Extrato para Gestão de Caixa	Extrato para Gestão de Caixa		F (Obrigatório) I (Opcional)
Empréstimo por Consignação	Empréstimo por Consignação	H (Obrigatório)	H (Obrigatório)
Pagamentos de Tributos	Pagamento de Contas e Tributos com Código de Barras	O (Obrigatório) W* (Opcional) Z (Opcional) B (Opcional) * Obrigatório para o pagamento de FGTS, convênios 0181 e 0182	O (Obrigatório) W* (Opcional) Z (Opcional) B (Opcional)
	Pagamento de Tributos sem Código de Barras	N (Obrigatório) B (Opcional) W (Opcional) Z (Opcional)	N (Obrigatório) B (Opcional) W (Opcional) Z (Opcional)
Consulta de Tributos a Pagar. A utilização desse serviço deverá ser previamente acordada com o banco.			N (Obrigatório)
Compor	Compor / Compor Rotativo	A (Obrigatório) B (Opcional) C (Opcional) I (Obrigatório) ou J (Opcional) I (Obrigatório)	A (Obrigatório) B (Opcional) C (Opcional) I (Obrigatório) ou J (Opcional) I (Obrigatório)

Tipos de *layouts* de registros detalhe.

24

4.5. Registro do tipo "trailer de lote"

A linha de trailer de lote é a linha que finaliza um lote de um tipo específico de serviço bancário. Abaixo dela aparecerão novos headers de lote ou o trailer do arquivo.

Assim como o trailer de arquivo, este registro serve para controle do lote.

Nesta linha estão presentes informações sobre:

- **Controle do arquivo**: informações sobre o banco (o mesmo da linha de header de arquivo), o número do lote e o identificador da linha (que para header de lote sempre será igual a 5);
- **Totais de registros e de valores** do lote;
- Outras informações complementares.

Um resumo do *layout* completo do registro **trailer de lote** você vê na tabela abaixo:

CAMPO				POSIÇÃO		Nº DIG	Nº DEC
				DE	ATÉ		
01.5	Controle	Banco	Código do Banco na Compensação	1	3	3	-
02.5		Lote	Lote de Serviço	4	7	4	-
03.5		Registro	Tipo de Registro	8	8	1	-
04.5	CNAB		Uso Exclusivo FEBRABAN / CNAB	9	17	9	-
05.5	Totais	Qtde. de Registros	Quantidade de Registros do Lote	18	23	6	-
06.5		Valor	Somatória dos Valores	24	41	16	2
07.5		Qtde. de Moeda	Somatória de Quant. de Moedas	42	59	13	5
08.5	Número Aviso Débito		Número Aviso de Débito	60	65	6	-
09.5	CNAB		Uso Exclusivo FEBRABAN / CNAB	66	230	165	-
10.5	Ocorrências		Códigos das Ocorrências p/ Retorno	231	240	10	-

***Layouts* da linha de trailer de lote.**

25

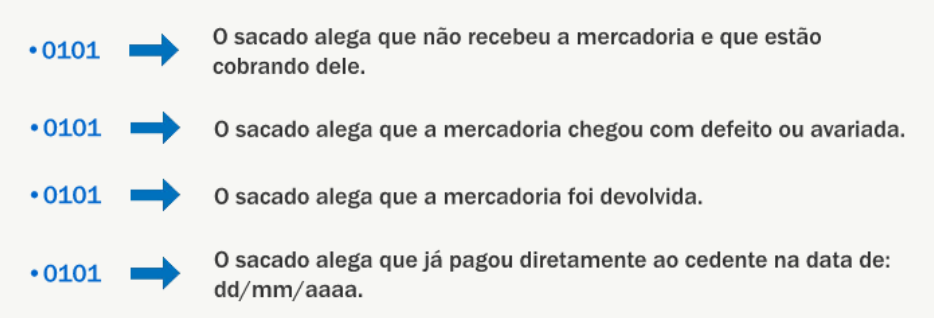
5 - CONTROLE DE ERROS

O CNAB utiliza vários campos nos registros detalhe de retorno e trailers de lote e de arquivo para controle de erros de operação. Vamos ver alguns exemplos:

5.1. Exemplo 1 – Registro detalhe – Segmento U

Para títulos de cobrança, há o envio de um registro detalhe do tipo Q (no arquivo de remessa). Para todos os títulos, o banco deverá devolver um outro arquivo (retorno). Esse arquivo de retorno conterá um registro do tipo Q para cada registro do tipo U processado.

Ocorrendo algum erro no processamento do título, haverá no registro de retorno tipo Q, no campo "código da ocorrência", que fica entre a posição 154 a 157 da linha, a informação do ocorrido. São exemplos de códigos que anunciam erro no processamento:



Todos esses motivos acima indicam que a cobrança não será feita, portanto **cancelada**. A empresa receberá essa informação no arquivo retorno a fim de poder, posteriormente, entrar em contato com o cliente para sanar dúvidas. Sendo assim, todas as informações de retorno devem poder ser analisadas pelos operadores do sistema da empresa, de modo que seja possível identificar problemas e tomar ações corretivas.

26

5.2. Exemplo 2 – Registro detalhe – Segmento T

No caso de títulos de cobrança, há ainda o envio de um registro detalhe do tipo P (no arquivo de remessa). Para todos os títulos, o banco deverá devolver um outro arquivo (retorno). Esse arquivo de retorno conterá um registro do tipo P para cada registro do tipo T processado.

Ocorrendo algum erro no processamento do título, haverá no registro de retorno tipo T, no campo "motivo da ocorrência", que fica entre a posição 214 a 223 da linha, a informação do ocorrido. São exemplos de códigos que anunciam erro no processamento:

- 01 = Código do Banco Inválido.
- 02 = Código do Registro Detalhe Inválido.
- 03 = Código do Segmento Inválido.
- 04 = Código de Movimento Não Permitido para Carteira.
- 05 = Código de Movimento Inválido.
- 06 = Tipo/Número de Inscrição do Cedente Inválidos.
- 07 = Agência/Conta/DV Inválido.

Todos esses motivos acima indicam que a cobrança não será feita, portanto **cancelada**. A empresa receberá essa informação no arquivo retorno com a finalidade de, posteriormente, poder entrar em contato com o cliente para sanar dúvidas. Sendo assim, todas as informações de retorno devem poder ser analisadas pelos operadores do sistema da empresa, de modo que seja possível identificar problemas e tomar ações corretivas.

Há inúmeros outros campos para registros de problemas. Consulte o manual do CNAB 240 para uma análise completa.

27

5.3. Contador de registros e de valores

Conforme já informado quando falamos de registros de trailer de reader e de arquivo, esses registros totalizam a quantidade de linhas do arquivo, quantidade de itens detalhe por lote, valor total do processamento do arquivo e valor subtotal do processamento de cada lote.

É de boa praxe que os desenvolvedores de sistemas utilizem essas informações para certificar que nenhuma linha foi inserida, alterada ou excluída.

Isso é feito pela contagem à parte via programação dos mesmos valores e quantitativos. O seu sistema, durante o processamento das linhas de registro detalhe, deve guardar em variáveis de memória os subtotais, totais e quantitativos de registros. Esses valores devem ser comparados com os valores informados nos registros de trailer de lote e de arquivo. Havendo qualquer divergência encontrada, significa que provavelmente o arquivo transmitido foi alterado ou corrompido. Isso pode acontecer por problemas de comunicação de rede, por exemplo. Portanto, se houve divergência de valores, todas as operações devem ser canceladas.

O sistema deve ser capaz de desfazer operações, o que em banco de dados chamamos de “Roll-back” (voltar para trás).

28

RESUMO

Neste módulo, aprendemos:

a. Que a Febraban é um órgão que define padrões bancários de tecnologia, entre os quais, o CNAB 240 que é um padrão utilizado para troca de dados bancários entre instituições, principalmente as financeiras.
b. Que o CNAB é um modelo baseado em arquivos de texto puro, com campos em posições fixas e largura de colunas com 240 caracteres.
c. Que cada linha de um arquivo CNAQ refere-se a um tipo específico de informação.
d. Que o padrão do CNAQ permite realizar 14 tipos de transações bancárias.
e. Que as informações estão descritas dentro de cada linha em posições fixas.
f. Que o modelo sempre prevê dois arquivos, um de remessa e outro de retorno.
g. Que o papel do arquivo de retorno é confirmar que as operações ocorreram com sucesso.
h. Que os tipos de linha são: header de arquivo, trailer de arquivo, header de lote, trailer de lote e linhas detalhe. Além dessas temos informações iniciais e finais de lote que são linhas opcionais.
i. Que as linhas representam registros e têm posições específicas dentro do arquivo.
j. Que é o oitavo caractere de cada linha que define que tipo de linha (registro) ela é.

k. Que as linhas de trailer devem ser utilizadas para testes de validação para saber se o arquivo não foi corrompido durante a transmissão.
l. Que o CNAB utiliza a tecnologia de texto puro para poder ser compatível com tecnologias mais antigas, como os computadores de grande porte, programas em Cobol e outros.

UNIDADE 2 – ARQUIVOS DE TEXTO, DOCUMENTOS, PLANILHAS E APRESENTAÇÕES

MÓDULO 4 – PLANILHAS, APRESENTAÇÕES E ARQUIVOS DIGITALIZADOS.

01

1 - ESCOPO E NÃO ESCOPO DESTE MÓDULO

Olá, seja bem-vindo a mais um módulo de estudos. Nas unidades passadas vimos conceitos básicos sobre como manipular arquivos baseados em texto, como um programa de computador pode manipular esses arquivos, e vimos também alguns tipos de arquivos como INI, LOG, XML, scripts e o padrão QNAB 240.

Neste módulo, iremos aprender algumas características sobre arquivos de planilhas, de apresentações e arquivos digitalizados. Mas, atenção, este módulo não pretende:

- Ensinar comandos básicos ou ser um minicurso de Word, Excel ou PowerPoint;
- Ser um minicurso de softwares livres de edição de documentos de texto, planilhas ou apresentações;
- Ensinar você a programar um sistema que manipula documentos.

Este módulo vai:

- Mostrar as características técnicas dos arquivos de texto, documentos, planilhas e apresentações, especialmente do ambiente MS Office.
- Apresentar as características dos documentos digitalizados especialmente aptos para um sistema SIGAD.

02

Atenção:

Para você mesmo fazer testes e verificar os exemplos citados neste módulo, seria interessante possuir o pacote Ms Office 2013 instalado no seu computador. Esse software, apesar de pago, pode ser obtido para avaliação por 30 dias a partir deste endereço:http://products.office.com/en-us/try ou deste: http://microsoft-office.softonic.com.br/. Uma alternativa é utilizar um software gratuito, como o OpenOffice, que pode ser baixado aqui:http://www.openoffice.org/download/ ou o LibreOffice, que pode ser baixado aqui: https://pt-br.libreoffice.org/. Todos os três softwares são chamados de “suítes

office", são semelhantes entre si e permitirão a você realizar os testes aqui descritos.

Outros exemplos deste módulo citam arquivos compactados; para esses exemplos seria interessante você possuir algum software de compactação e descompactação de arquivos como o Winzip (www.winzip.com), o Winrar (www.win-rar.com) ou o 7Zip (www.7-zip.org), este último gratuito.

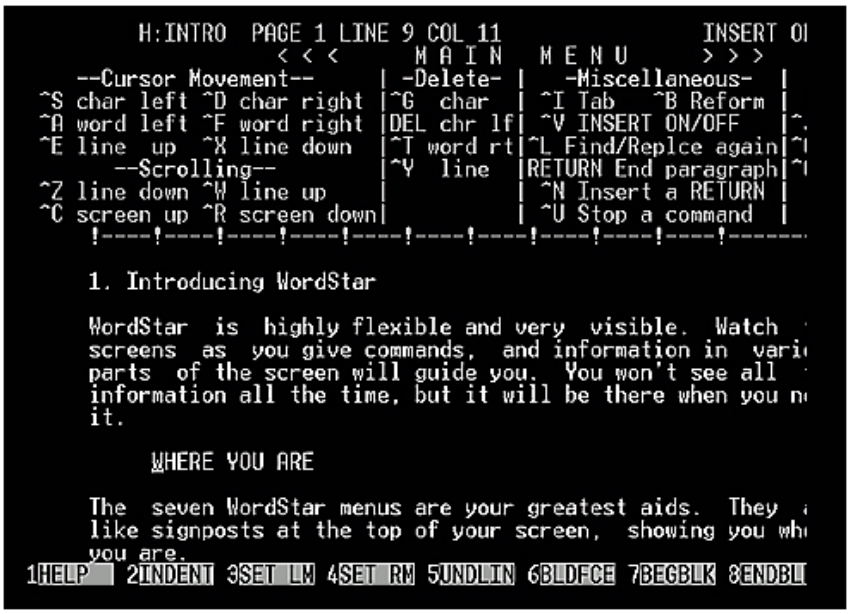
03

2 - DOCUMENTOS DE TEXTO

Arquivos de texto formatado existem desde a década de 70! Naquela época, as formatações que existiam eram:

- opção para margens de parágrafo (esquerda, direita, centralizado ou justificado),
- letras em negrito e letras em itálico.

Somente isso! Mesmo com tão poucas funcionalidades, os arquivos de texto já se tornavam populares e com um *layout* muito melhor que os arquivos de texto puro. Dois dos programas editores de texto mais famosos que existiram naquela época foram o WordStar e o WordPerfect. Naquela época não existia mouse ou telas gráficas, tínhamos apenas o espaço da tela representado por 80 caracteres de largura por 24 linhas de altura.



Tela padrão do WordStar.

De lá pra cá muita coisa mudou. Já temos diversas formatações disponíveis, como a escolha de fontes, tamanho de fontes, cores, formato dos caracteres e muito mais.

04

a) Padrão RTF

O primeiro tipo de arquivo que aceitou um padrão mais complexo de formatação foi o RTF (*Rich Text Format ou Formato de Texto Rico*). O RTF aceita texto formatado, tabelas e gráficos. Ele foi criado em 1987 pela Microsoft e, a partir de então, tornou-se o primeiro formato padrão para arquivos de texto com formatação. O primeiro editor de textos da Microsoft a suportar manipular documentos RTF foi chamado de Ms Write, desenvolvido em 1988.

Da mesma forma que a Febraban (Federação Brasileira de Bancos) definiu o padrão CNAB 240 para movimentações bancárias, a Presidência da República, no Manual de Redação da Presidência da República, definiu em 2002 que o RTF deve ser o padrão preferencial para textos compartilháveis entre organizações públicas. O STF (Supremo Tribunal Federal) e o TJ (Tribunal de Justiça) atualmente só emitem súmulas no padrão RTF.

A grande vantagem de se utilizar o padrão RTF para troca de arquivos de texto entre sistemas diferentes é que, como ele é um modelo estanque e aberto, há uma garantia de que o documento criado por um sistema e lido por outro sistema será exatamente igual ao documento lido e confeccionado no sistema original.

A codificação de um arquivo RTF é feita em arquivo de texto plano, utilizando delimitadores como a barra invertida ("\"), sinais de chaves ("{ }") e os sinais de maior que e menor que ("<...>"). Um pequeno texto no formato RTF fica assim armazenado dentro do arquivo RTF:

```
{\rtf1\ansi{\fonttbl\f0\fswiss Helvetica;}\f0\pard
Este é um pequeno exemplo de {\b texto} feito no formato RTF.\par
}
```

Observe que facilmente conseguimos ler que o conteúdo do arquivo é "Este é um pequeno exemplo de texto feito no formato RTF". Por isso chamamos de "arquivo próximo à leitura humana", diferentemente dos arquivos binários, que são "arquivos em linguagem de máquina". O texto em RTF acima coloca a palavra "texto" em negrito, por isso a marcação "{\b }". Ao aprender essas marcas, fica fácil saber o resultado final da formatação.

padrão RTF
O padrão técnico de criação de documentos RTF está disponível aqui: http://www.microsoft.com/en-us/download/confirmation.aspx?id=10725

05

Como a codificação é feita em arquivo de texto puro (não é binário, nem criptografado) podemos armazenar o conteúdo de um arquivo RTF dentro de um registro em uma tabela de um banco de dados.

Podemos ainda fazer pesquisas do tipo "SELECT" para analisar o conteúdo do arquivo à procura de palavras-chave. Isso é muito interessante para um SIGAD, por exemplo, que deve possuir funcionalidades de pesquisas por conteúdo dentro dos arquivos e documentos arquivísticos.

Por fim, é importante ressaltar que praticamente todos os editores de texto modernos suportam ler e gravar documentos no formato RTF.

Mais vantagens do padrão RTF:

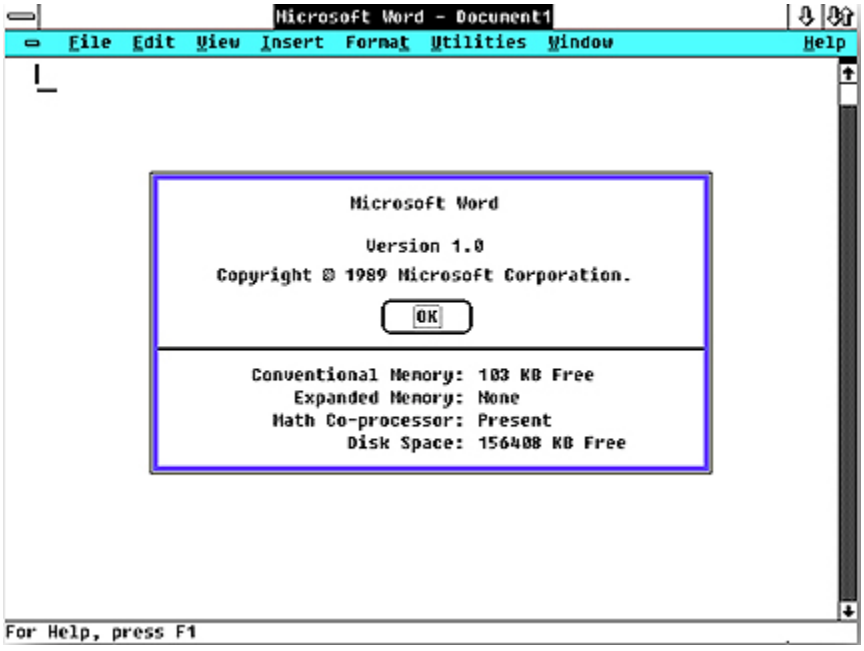
- Não é codificado em arquivo binário;
- Pode ser convertido para qualquer outro formato;
- Permite formatação de fonte e texto;
- Suporte gráficos e tabelas;
- Tem padrão aberto;
- Todos os editores de texto suportam RTF;
- Arquivos RTF podem ser guardados e salvos em Bancos de Dados facilmente.

06

b) Padrão DOC

Como já falamos em outros módulos sobre documentos de texto, vamos focar agora em algumas características técnicas.

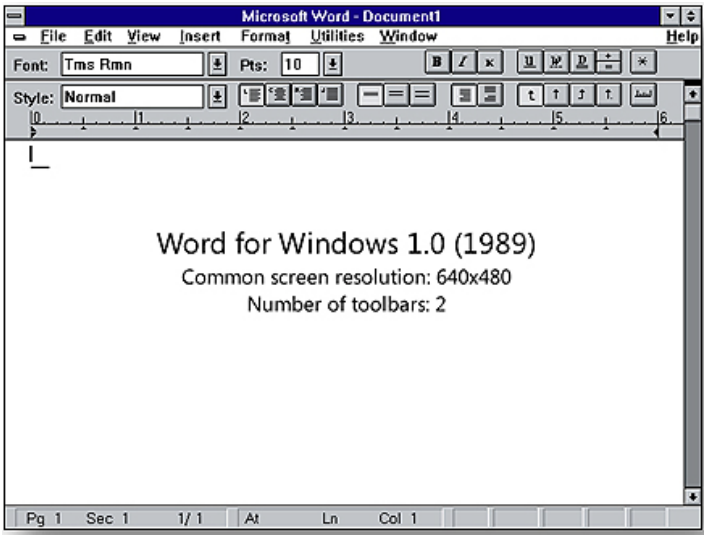
Após a década de 1980, a Microsoft se tornou a empresa líder em software para elaboração de documentos de texto. O Ms Word foi criado em 1981 pela Microsoft e comercializado em 1983. Nessa mesma época foi definida a extensão .DOC para arquivos de texto da Microsoft.



MS Word versão 1.0 para DOS

07

Em 1989 a Microsoft desenvolveu a sua primeira versão gráfica do editor de texto, denominado Ms Word for Windows, versão 1.0. Esse aplicativo funcionava no Windows 1.0.



Ms Word for Windows 1.0

08

Mas foi apenas em 1990, com a chegada do Ms Windows 3.0 que ocorreu o boom de crescimento da Microsoft e tanto seu sistema operacional como seu pacote Office popularizou-se no mundo todo.

A última versão do Ms Word que tinha como padrão documentos do tipo DOC foi a versão 2007. Da versão 1 até a versão 2007 houve uma evolução natural com acréscimo exorbitante de funcionalidades. Já na versão 2007, o Ms Word contabilizava mais de uma centena de funcionalidades, dentre as quais, as mais utilizadas são:

- controle de versões,
- revisão,
- arquivos inseridos dinamicamente,
- compactação de imagens,
- compartilhamento segmentado,
- controle de acesso e outros.

VANTAGENS DO PADRÃO DOC	DESVANTAGENS DO PADRÃO DOC
→ Padrão amplamente reconhecido. → Lido e interpretado pela maioria dos softwares concorrentes. → Possui uma infinidade de funcionalidades. → Altíssima qualidade e fácil de usar.	→ Arquivo binário. → Somente acessível por API específica. → Difícil de interpretar e indexar conteúdo sem API específica. → Padrão fechado, proprietário e pago. → Pode gerar problemas de corrupção de arquivo quando o arquivo é muito grande. → O arquivo não é compactado e fica muito grande, se comparado a outros formatos.

09

c) Padrão DOCX

O padrão DOCX surgiu com o Ms Word 2010 e hoje estamos com as versões Ms Word 2013 (que é instalada localmente e o Ms Word 365 (que é acessado 100% via web). É interessante lembrar esse novo conceito de software pago: você pode pagar por uma licença de uso permanente (modelo padrão – Office 2013) ou pagar mensalmente pelo uso (ideia de aluguel de software – Office 365).

Outra importante funcionalidade do segundo modelo é que seu software e seus arquivos ficam disponíveis em qualquer lugar da Internet, ou seja, onde quer que você esteja, de qualquer micro ou dispositivo móvel com acesso à Internet, você poderá ver e editar seus arquivos.

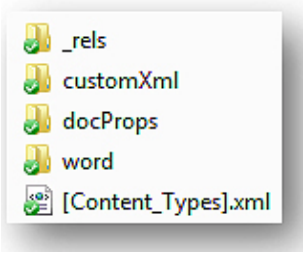
Além disso, o Office 365 permite o compartilhamento de arquivos (você diz quem pode ler e quem pode editar) e permite também a edição simultânea: duas pessoas trabalhando em partes diferentes do arquivo. Se necessário, você pode salvar o arquivo localmente nos diversos padrões suportados pelo Ms Word. Voltando a falar do padrão DOCX, ele é uma forma compactada de informação em XML sobre os documentos.

Veja que interessante: o Word (e demais produtos do Office) agora não utilizam mais informações binárias, mas sim, informações em XML! Com isso, temos acesso total ao conteúdo dos arquivos, fica muito mais fácil manipular diretamente os arquivos, sem a necessidade de componentes interpretadores (vamos exemplificar mais adiante).

10

- **Conteúdo de um arquivo DOCX**

Primeiramente, vamos fazer um teste prático: crie um documento no formato DOCX. Salve esse documento como “teste.docx”. Agora, renomeie o arquivo para “teste.zip”. Abra esse arquivo com o Explorer, o Winzip ou outro programa de compactação de arquivos. Descompacte o arquivo para uma pasta temporária. Você verá que o arquivo “teste.docx” se tornou o conteúdo abaixo:



Conteúdo básico de um arquivo DOCX

Observe agora o conteúdo de cada pasta e cada arquivo:

- A pasta "**_rels**" contém o arquivo ".rels" que é um arquivo XML que descreve que é utilizados padrões XML abertos.
- A pasta "**customXml**" contém arquivos XML que especificam componentes e versões de XML utilizados no documento e a subpasta "_rels" com informação sobre padrões XML utilizados.
- A pasta "**docProps**" contém dois arquivos:
  - O arquivo "app.xml"
  - O arquivo "core.xml".

Veremos a pasta "**word**" a seguir.

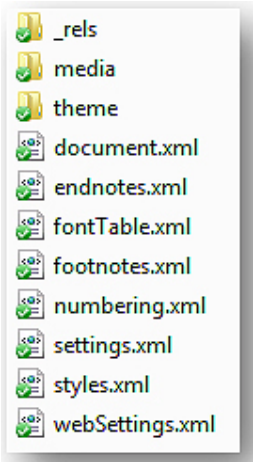
app.xml
O arquivo "app.xml" apresenta algumas propriedades do documento, como: tipo de arquivo, título do documento, quantidade de caracteres, de palavras, de linhas e de páginas, o tempo total em segundos que o documento ficou sendo digitado e outras informações.

core.xml
O arquivo "core.xml" apresenta outras propriedades do documento, como: o título, o autor do documento, quem o modificou por último, quantas vezes foi salvo a data/hora de criação e a data/hora da última versão.

11

- A pasta "word", que é a mais importante de todas para nós, contém:
  - A subpasta "_rels", que contém padrões XML adotados;
  - A pasta "media" que contém todos os objetos de imagem e som utilizados do documento (todos os objetos ficam com a extensão ".tmp", mas no fundo são imagens no formato ".PNG" e arquivos de som no formato ".MP3", você pode renomear esses arquivos para ver o conteúdo deles).
  - A pasta "theme" que contém informações sobre o tema utilizado no documento, como: fontes, tamanhos, padrões internacionais de idioma, e outros.

- O arquivo “document.xml” que contém todas as informações digitadas no documento (todo o texto), e por isso é considerado o principal arquivo XML do conjunto.
- O arquivo “endnotes.xml” que contém informações sobre notas e separadores de página.
- O arquivo “fontTable.xml” que contém informações sobre as fontes utilizadas no documento.
- O arquivo “footnotes.xml” que contém informações sobre notas de rodapé e notas.
- O arquivo “numbering.xml” que contém informações sobre o posicionamento e dimensão dos parágrafos, imagens e objetos.
- O arquivo “settings.xml” que contém informações sobre configurações de usuário utilizadas no documento.
- O arquivo “styles.xml” que contém informações sobre os estilos utilizados (título, subtítulo, parágrafo, etc).
- E, finalmente, o arquivo “webSettings.xml” que contém informações sobre o layout do documento caso ele seja salvo no formato HTML (caso ele seja apresentado em um navegador WEB).



Conteúdo da pasta “word”

12

Apresentamos a seguir algumas vantagens e desvantagens do padrão DOCX:

VANTAGENS DO PADRÃO DOCX	DESVANTAGENS DO PADRÃO DOCX
→ Conteúdo em padrão aberto baseado em XML. → Permite a pesquisa e indexação de conteúdo sem API específica. → Arquivo compactado, bem pequeno se comparado a outros formatos. → Possui uma infinidade de funcionalidades. → Altíssima qualidade e fácil de usar.	→ O padrão DOCX não é amplamente reconhecido. → Não pode ser lido e interpretado pela maioria dos softwares concorrentes. → Precisa ser descompactado antes da interpretação do conteúdo XML.

13

2.1 - Como um programa pode manipular um arquivo DOC ou DOCX

As **manipulações** mais comuns que um programa pode fazer em um documento de texto são:

- Ler todo o conteúdo de um documento para salvá-lo dentro do banco de dados a fim de possibilitar pesquisas sobre o documento. Exemplo: Em um sistema SIGAD, ler o conteúdo e guardá-lo em tabelas de pesquisa.
- Criar documentos a partir de modelos pré-concebidos. Exemplo: suponha que você trabalhe em uma Imobiliária e você tenha um modelo de documento para elaborar um Contrato de Aluguel. A partir das informações cadastradas no sistema sobre o dono do imóvel, os dados do imóvel, os dados do locatário e os dados da locação, o sistema poderia produzir automaticamente o contrato com todas as informações pré-cadastradas.
- Criar documentos automaticamente, a partir de conteúdo pré-definido no banco de dados. Exemplo: suponha que sua empresa possua uma base de dados com o nome e endereço de 1.000 clientes e que ela queira mandar uma mensagem de Natal padronizada para todos os clientes. O sistema poderia ler os dados de todos os clientes e a partir de um modelo de documento, criar e imprimir todas as 1.000 cartinhas automaticamente.

Quanto à programação em si, lembre-se de quando falamos de arquivos INI e XML, que era muito mais fácil manipular os arquivos por meio de componentes do que diretamente (como fizemos em arquivos TXT), lendo linha a linha.

14

A forma mais fácil de um programa moderno manipular documentos DOC ou DOCX é por meio de componentes Office. Os componentes Office são gratuitos e podem tanto ser instalados manualmente (baixando-os do site da Microsoft) quanto instalados automaticamente (quando você instala o Office no seu computador).

Quando você instala um componente de uma determinada versão, ele sempre será compatível com aquela versão e todas as anteriores, portanto, um componente versão 2010 consegue manipular um documento feito, por exemplo, nas seguintes versões do Word: 1.0, 3.0, XP, 2000, 2003 e 2007.

Um componente de documento Word permite, entre outras funcionalidades:

- Abrir um documento Word guardado em uma pasta em qualquer dispositivo local ou remoto;
- Abrir um documento Word salvo na Internet;
- Abrir um documento Word salvo em banco de dados ou no SharePoint (sistema de Sites da Microsoft);
- Criar um documento em branco ou baseado em um modelo;
- Inserir, alterar ou excluir texto;
- Alterar formatos e estilo (fontes, tamanho, sublinhado, cores, negrito etc.);
- Inserir, alterar ou excluir tabelas;
- Inserir, alterar ou excluir imagens e outros objetos;
- Imprimir documentos;
- Salvar o arquivo entre os vários formatos suportados (DOCX, DOC, RTF, ODT etc.);

15

2.1.1 - Exemplo de trecho de código fonte em Visual Basic dot Net para manipulação de documento Word (doc ou docx)

Cabe salientar que não é escopo do exemplo a seguir ensinar a você programar em Visual Basic dot Net. O trecho do programa abaixo serve apenas para lhe mostrar como o uso de componentes de uma linguagem de programação facilita a manipulação de arquivos.

Você aprenderá a programar na matéria específica de programação. Independente da linguagem de programação, Visual Basic, Java, C# ou outra, a manipulação dos componentes será a mesma, somente o formato e a sintaxe muda de uma linguagem de programação para outra. Os conceitos apresentados abaixo lhe serão muito úteis quando você for criar um software que manipula documentos no formato Ms Word.

a) Para fazer o seu programa referenciar o componente Ms Word, é utilizado o seguinte código:

```
Imports Word = Microsoft.Office.Interop.Word
```

Esse comando diz para seu programa “importe todo o conjunto de funcionalidades do componente Ms Word para dentro do meu programa.

b) Seu programa precisará de variáveis para receber e controlar as informações relacionadas ao documento. Para isso, são necessárias as seguintes definições de variáveis para esse exemplo:

```
Dim oWord As Word.Application       ‘ Componente principal para manipular o documento.
Dim oDoc As Word.Document           ‘ O próprio documento em si.
Dim oTable As Word.Table            ‘ Uma tabela que criaremos dentro do documento.
Dim oPara1 As Word.Paragraph, oPara2 As Word.Paragraph  ‘ Três parágrafos do documento.
Dim oPara3 As Word.Paragraph
Dim oRng As Word.Range                  ‘ Um trecho qualquer do documento.
Dim oShape As Word.InlineShape          ‘ Uma forma que criaremos no documento.
Dim oChart As Object                    ‘ Um gráfico que criaremos no documento.
Dim Pos As Double                       ‘ A posição exata onde criaremos os objetos.
```

16

c) Com as variáveis definidas, vamos agora criar um documento em branco:

```
'Inicia o Word e abre um novo documento.
oWord = CreateObject("Word.Application")        ‘ Abre o word.
oWord.Visible = True                            ‘ Diz que ele estará visível para o
usuário.
oDoc = oWord.Documents.Add                      ‘ Cria um documento em branco.
```

d) Com o documento novo aberto, vamos criar um parágrafo:

```
'Insere um parágrafo no início do documento.
oPara1 = oDoc.Content.Paragraphs.Add        ‘ Cria um objeto do tipo parágrafo.
oPara1.Range.Text = "Este é o cabeçalho do documento"   ‘ Define um texto para ele.
oPara1.Range.Font.Bold = True                           ‘ Define que a fonte será em
negrito.
oPara1.Format.SpaceAfter = 24    ‘ Define que haverá 24 pontos de espaço entre este e o
próximo parágrafo.
oPara1.Range.InsertParagraphAfter()   ‘ Diz para inserir este parágrafo após a posição
atual do cursor.
```

e) Vamos agora inserir dois parágrafos no final do documento (após o primeiro parágrafo criado no passo anterior):

```
'Insere um parágrafo no final do documento.
'** \endofdoc indica que é no final do documento que iremos inserir o novo parágrafo.
oPara2 = oDoc.Content.Paragraphs.Add(oDoc.Bookmarks.Item("\endofdoc").Range)    ‘ Indica o final.
oPara2.Range.Text = "Este texto está no final do documento"                     ‘ Texto.
oPara2.Format.SpaceAfter = 6                ‘ Define o espaçamento.
oPara2.Range.InsertParagraphAfter()         ‘ Insere o parágrafo.

'Insere outro parágrafo.
oPara3 = oDoc.Content.Paragraphs.Add(oDoc.Bookmarks.Item("\endofdoc").Range)
oPara3.Range.Text = "Abaixo deste parágrafo iremos inserir uma tabela:"
oPara3.Range.Font.Bold = False
oPara3.Format.SpaceAfter = 24
oPara3.Range.InsertParagraphAfter()
```

17

f) Agora iremos inserir uma tabela com 3 linhas e 5 colunas:

```
'Insere uma tabela e preenche ela com dados.
Dim r As Integer, c As Integer      ‘Duas variáveis de apoio para controlar linhas e colunas.
oTable = oDoc.Tables.Add(oDoc.Bookmarks.Item("\endofdoc").Range, 3, 5)  ‘ Cria a tabela.
oTable.Range.ParagraphFormat.SpaceAfter = 6     ‘ Define o espaçamento.
For r = 1 To 3          ‘ Laço para percorrer todas as linhas.
    For c = 1 To 5      ‘ Laço para percorrer todas as colunas.
            oTable.Cell(r, c).Range.Text = "Linha: " & r & "Coluna: " & c   ‘ Preenche com
informações
sobre linhas e colunas.
            Next
        Next
        oTable.Rows.Item(1).Range.Font.Bold = True          ‘ Define que a primeira célula será
em negrito.
        oTable.Rows.Item(1).Range.Font.Italic = True        ‘ Define que a primeira célula será
em itálico.
```

g) Agora vamos inserir um gráfico:

```
'Insere um gráfico.
oShape =
oDoc.Bookmarks.Item("\endofdoc").Range.InlineShapes.AddOLEObject(ClassType:="MSGraph.Chart.8")
oChart = oShape.OLEFormat.Object
oChart.charttype = 4 'xlLine = 4        ‘ Diz que é um gráfico de linhas
oChart.Application.Update()
oChart.Application.Quit()
```

h) Pronto, finalizamos a edição do arquivo. Vamos agora salvar e fechar o documento:

```
oDoc.Save       ‘ Salva o documento.
oWord.Quit      ‘ Fecha o word.
```

i) Fim!

Neste link você encontrará informações completas de como um programa manipula um documento em Word:http://support.microsoft.com/kb/316383/pt-br.

Como você pôde observar, há uma série de componentes e objetos que criam os itens de um documento. Por meio da manipulação dos componentes, você consegue fazer um programa que manipula todo o documento.

18

2.2 - Alguns outros tipos de arquivos criados pelo Ms Word:

Além dos já citados .txt, .rtf, .doc, .docx, o Ms Word cria documentos em outros formatos de arquivos interessantes, vamos ver alguns deles:

EXTENSÃO:	O QUE É:
.docm	Arquivo com macros habilitadas (permitem automação de tarefas)
.dotx	Arquivo de modelo, quando criado ele não é editado, mas sempre um novo documento é criado a partir dele.
.dotm	Arquivo de modelo com macros habilitadas.
.dot	Arquivo modelo no formato antigo 97-2003.
.pdf	Arquivo apenas para leitura no formato PDF, comumente utilizado para compartilhamento na Internet e em sistemas SIGAD para documentos digitalizados.
.xps	Arquivo em formato semelhando ao PDF, porém é um padrão proprietário da Microsoft.
.mth	Arquivo em formato de página WEB para arquivo único.
.htm ou .html	Arquivo no formato de página WEB, porém com imagens e formatos descritos em outros arquivos.
.xml	Arquivo em formato xml.
.odt	Arquivo no formato Open Office (formato aberto).

19

2.3 - Padrão para ODT – OpenDocument ou Documento Aberto

Assim como existe o pacote de escritório (Office) da Microsoft que incorpora o Ms Word como editor de texto, existe também pacotes de escritório gratuitos, como o Apache OpenOffice e o LibreOffice.

Todos os pacotes Office citados são compatíveis com o formato ODF (Open Document Format). Os documentos ODF, assim como a nova versão dos programas Ms Office, utilizam arquivos compactados no formato XML. Os novos formatos são muito semelhantes, dessa forma, o que falamos anteriormente sobre o DOCS se aplica quase que totalmente para o ODT.

Assim como o formato XLSX é semelhante ao formato ODS e o formato PPTX semelhante ao formato ODP. Na verdade, todos esses arquivos são semelhantes entre si, são apenas atributos internos de formatação que define se o documento é um texto, uma planilha ou uma apresentação.

As principais extensões para os arquivos ODF são:

- ODT (documentos de texto) – semelhante ao formato DOCX do Ms Word;
- ODS (folhas de cálculo) – semelhante ao formato XLSX do Ms Excel;
- ODP (documentos de apresentação) – semelhante ao formato PPTS do Ms PowerPoint.

A vantagem do ODF é que ele não está vinculado a nenhum conjunto de programas de escritório. É um padrão aberto que qualquer empresa pode implementar nos seus programas. O OpenOffice usa o formato ODF como formato padrão de documentos. A maioria dos outros processadores de texto modernos também tem a capacidade de importar e exportar arquivos no formato ODF.

O formato ODF foi desenvolvido por uma grande variedade de organizações, sendo possível acessar livremente as respectivas especificações. Isto significa que o ODF pode ser implementado em qualquer

sistema, seja ele de código aberto ou não, sem ser necessário efetuar qualquer tipo de pagamento ou estar sujeito a uma licença de uso restrito. Saiba+

ODF
ODF (Open Document Format ou documento de formato aberto) é um formato padrão internacional ISO para documentos de escritório, criado em 2006.

Saiba+
O ODF constitui-se uma alternativa às formas de documentação que são propriedade de empresas privadas, sujeitos a licença de uso restrito ou onerosas, permitindo a organizações e indivíduos escolherem as aplicações para escritório que mais lhes convêm para lidar com os arquivos guardados que o ODF lhes oferece. O formato é independente de plataforma e fornecedor, tornando-o adequado ao arquivo de documentos em longo prazo.

20

3 - PLANILHAS

Planilhas são como tabelas. A principal vantagem do uso de planilhas é a automação de cálculos que podem ser feitos. O mais famoso editor de planilhas do mercado é o Ms Excel. Se você não conhece nada sobre planilhas, aconselho a você a olhar alguns vídeos sobre o Ms Excel.

Quantidade	Produto	Unitário	Subtotal
2	Lata de Óleo	R$ 2,34	R$ 4,68
12	Leite	R$ 3,34	R$ 40,08
4	Sabonete	R$ 0,88	R$ 3,52
2	Pão de Forma	R$ 5,54	R$ 11,08
0,5	Carne Patinho	R$ 23,10	R$ 11,55
		TOTAL	**R$ 70,91**

Exemplo de planilha com valores de subtotal e total calculadas automaticamente.

A história de evolução do Excel é muito semelhante à do Word. Afinal, ambos foram criados juntos e assim vêm evoluindo.

Word, **Excel**, e também, o **Access** (para banco de dados), o **Outlook** (para e-mail e agenda), o PowerPoint (para apresentações) e outros aplicativos são comercializados juntos num pacote denominado **Ms Office**.

Dessa forma, as características, vantagens e desvantagens e manipulação de arquivos relacionados às planilhas são as mesmas daquelas já vistas relacionadas aos documentos de texto.

Vídeos
Você pode encontrar alguns desses vídeos acessando este link: http://goo.gl/5lj4ke.

21

3.1 - Exemplo de trecho de código fonte em Visual Basic dot Net para manipulação de planilha Excel (xls ou xlsx)

Mais uma vez ressaltamos que não é escopo do exemplo a seguir ensinar a você programar em Visual Basic dot Net. O trecho do programa abaixo serve apenas para lhe mostrar como o uso de componentes de uma linguagem de programação facilita a manipulação de arquivos.

Você aprenderá a programar na matéria específica de programação. Independente da linguagem de programação, Visual Basic, Java, C# ou outra, a manipulação dos componentes será a mesma, somente o formato e a sintaxe muda de uma linguagem de programação para outra. Os conceitos apresentados abaixo lhe serão muito úteis quando você for criar um software que manipula planilhas no formato Ms Excel.

a) Para fazer o seu programa referenciar o componente Ms Office, é utilizado o seguinte código:

```
Imports Microsoft.Office.Core
```

Esse comando diz para seu programa “importe todo o conjunto de funcionalidades do componente Ms Office para dentro do meu programa.

b) Seu programa precisará de variáveis para receber e controlar as informações relacionadas ao documento. Para isso, são necessárias as seguintes definições de variáveis para esse exemplo:

```
Dim oXL As Excel.Application    ‘ Componente principal para manipular o excel.
Dim oWB As Excel.Workbook       ‘ Componente do tipo conjunto de planilhas.
Dim oSheet As Excel.Worksheet   ‘ Componente do tipo aba de planilha.
Dim oRng As Excel.Range         ‘ Componente que aponta para uma posição do cursor ou
um trecho marcado.
```

22

c) Inicializa o Excel de dentro do programa:

```
oXL = CreateObject("Excel.Application")     ‘ Abre o Excel.
oXL.Visible = True                          ‘ Torna-o visível para o usuário.
```

d) Cria um conjunto de planilhas (workbook):

```
oWB = oXL.Workbooks.Add     ‘ Adiciona um conjunto de planilhas.
oSheet = oWB.ActiveSheet    ‘ Torna-o ativo, referenciando o objeto oSheet à
primeira planilha.
```

e) Adiciona texto nas 4 primeiras células da planilha:

```
oSheet.Cells(1, 1).Value = "Nome:"
oSheet.Cells(1, 2).Value = "Idade:"
```

```
oSheet.Cells(1, 3).Value = "Endereço:"
oSheet.Cells(1, 4).Value = "Telefone:"
```

f) Formata as células A1 a D1 como negrito e faz o alinhamento do texto ser centralizado:

```
With oSheet.Range("A1", "D1")   ‘ Para as células A1 a D1…
            .Font.Bold = True       ‘ Coloca o texto em negrito.
            .VerticalAlignment = Excel.XlVAlign.xlVAlignCenter  ‘ Alinha o texto no
centro.
End With
```

23

g) Cria uma matriz e adiciona valores:

```
Dim saNames(1, 3) As String     ‘ Matriz 2 x 4 (lembre-se que o primeiro item é o
zero)
saNames(0, 0) = "Marcelo"
saNames(0, 1) = "43"
saNames(0, 2) = "Rua das couves,18, apartamento 1091"
saNames(0, 3) = "8785-4785"
saNames(1, 0) = "Joana"
saNames(1, 1) = "18"
saNames(1, 2) = "Rua das Palmeiras, 38"
saNames(1, 3) = "4457-8754"
```

h) Preenche as células com os valores da matriz:

```
oSheet.Range("A2", "D3").Value = saNames        ‘ Aponta a Matriz para o espaço
delimitado.
```

i) Garante que a planilha está visível e, depois, fecha a planilha.

```
oXL.Visible = True          ‘ Mostra a planilha
oXL.UserControl = True      ‘ Mostra os controles da planilha
oXL.Quit()                  ‘ Fecha o excel.
```

j) Fim!

Neste link você encontrará informações completas de como um programa manipula uma planilha em Excel:http://support.microsoft.com/kb/301982/pt-br.

24

3.2. Alguns outros tipos de arquivos criados pelo Ms Excel

Assim como Ms Word cria documentos em outros formatos de arquivos interessantes, o Ms Excel também cria arquivos em outros formatos. Vamos ver alguns deles:

EXTENSÃO:	O QUE É:
.xlsm	Planilha com macros habilitadas (permitem automação de tarefas)
.xlsb	Planilha em formato binário (para esconder as informações originais)
.xltx	Planilha modelo, quando criado ele não é editado, mas sempre uma nova planilha é criada a partir dele.
.xltm	Planilha modelo com macros habilitadas.
.txt	Texto puro em formato plano ou separado por tabulações.
.cvs	Texto puro em formato plano separado por vírgulas.
	Planilha apenas para leitura no formato PDF, comumente utilizado para compartilhamento na Internet e em sistemas SIGAD para documentos digitalizados.
.pdf	Planilha em formato semelhando ao PDF, porém é um padrão proprietário da Microsoft.
.xps	Planilha em formato de página WEB para Planilha único.
.mth	Planilha no formato de página WEB, porém com imagens e formatos descritos em outras Planilhas.
.htm ou .html	Planilha em formato xml.
.xml	Planilha no formato Open Office (formato aberto).

Assim como para documentos de texto em formato aberto existe o padrão ODT, no caso de planilhas, a extensão utilizada é a **ODS**. Mais detalhes já foram abordados anteriormente neste módulo.

25

4 - Apresentações

Apresentações são como slides ou fotos utilizadas muitas vezes para demonstrar uma ideia. Professores, vendedores e empresas utilizam apresentações em slides para expor suas ideias para um público definido. Normalmente vemos apresentações serem projetadas em projetores de multimídia quando temos muitas pessoas interessadas no assunto. Mas também podemos ter apresentações expostas no monitor para apenas um único usuário. Saiba+

A história do PowerPoint, aplicativo mais popular para elaboração de apresentações, é bem mais moderna que os editores de texto e planilha. Apresentações surgiram na década de 90, com a chegada dos monitores gráficos e das impressoras jato de tinta. Já os editores e planilhas surgiram 20 anos antes. Mesmo assim, hoje em dia, as características dos arquivos de apresentação, bem como a forma de manipular esses arquivos são sempre bem semelhantes.

4.1 Manipulação de uma planilha PowerPoint (ppt ou pptx).

Embora seja possível criar e manipular uma planilha em PowerPoint de maneira muito semelhante ao que vimos quando falamos de Word e Excel, na vida prática é raríssimo vermos programas que necessitam desse tipo de manipulação. Por ser tão raro, não vamos aqui expor um exemplo sobre isso. Entretanto, sendo do seu interesse analisar um código de um programa que manipula um arquivo PowerPoint, acesse este link: http://support.microsoft.com/kb/160822/pt.

Saiba+
Caso você não conheça nada sobre apresentações, sugiro que você assista a alguns desses vídeos: http://goo.gl/xpo3Io.

26

5 - ARQUIVOS DIGITALIZADOS

Arquivos digitalizados se referem a documentos em papel que depois de um processo de digitalização se tornaram documentos eletrônicos.

O mais comum de vermos são documentos do tipo texto no formato de papel A4 ou documentos menores, como uma identidade, um cartão de CPF, uma carteira de motorista ou outro, que é passado por um scanner e transformado em documento digitalizado. Entretanto há scanners especiais que podem digitalizar livros, mapas, plantas e outros documentos em grandes formatos.

Ainda, recentemente, temos visto a digitalização por meio de máquinas fotográficas. Sim, fotos em arquivos digitais são representações eletrônicas de imagens e documentos.

Os equipamentos mais utilizados para digitalização são:

• **Scanners**: podem ser de mesa, móveis, embutidos em impressoras multifuncionais, específicos para livros, específicos para grandes formatos.

• **Fotografias**: podem ser máquinas fotográficas profissionais ou mesmo um celular.

27

Há várias formas de transformar um documento ou algo físico em formato digital, vamos ver as características dos principais formatos:

NOME DO FORMATO	PRINCIPAL USO	VANTAGENS	DESVANTAGENS
Arquivo PDF	Documentos em papel digitalizados em scanners.	• Pode conter várias páginas. • É o padrão mundial mais bem aceito. • É compatível com SIGAD. • Pode ter o conteúdo indexado. • Pode ter o conteúdo protegido por senha.	• Precisa de um outro programa para abrir (como o Adobe Reader).
Arquivo TIFF	Documentos em papel digitalizados em scanners.	• Tem altíssima qualidade. • Muito bom para digitalizar fotos e imagens. • Pode conter várias páginas	• Arquivo muito grande. • Difícil manipulação. • Precisa de software de imagem para ver o arquivo.
Arquivo JPG	Fotografias	• Produz arquivos pequenos. • Alta compatibilidade com sistemas. • Apropriado para fotos e imagens.	• Difícil de identificar o texto para indexação. • Não é um padrão reconhecido para SIGAD. • Não é apropriado para documentos de texto. • Se houver muita compactação, perde a qualidade da imagem. • Não suporta multipáginas. • Arquivos muito grandes. • Não suporta multipáginas
Arquivo RAW	Fotografias feitas em máquinas fotográficas avançadas e profissionais	• Altíssima qualidade	• Difícil compatibilidade. • Precisa de editor de imagens específico para ver o conteúdo.

Dos formatos apresentados, arquivos PDF são de longe os mais utilizados para a digitalização de documentos.

28

5.2 - Como manipular documentos digitalizados.

A manipulação de documentos digitalizados é algo bastante complexo. Dependendo do objetivo do seu programa, essas seriam algumas das operações possíveis em documentos digitalizados:

- Extrair o conteúdo do arquivo para análise e indexação;
- Converter o conteúdo do arquivo em um documento de texto;
- Extrair partes das informações como campos pré-definidos ou imagens.

5.3 - Padrões de arquivos PDF e TIFF digitalizados

Há várias formas de você configurar um scanner para produzir um arquivo PDF ou TIFF. Entre as opções mais importantes temos:

A) QUANTO AO TIPO DE IMAGEM	B) QUANTO À RESOLUÇÃO
O tipo de imagem está relacionado com a quantidade de cores que o documento digitalizado irá produzir. Quando maior a quantidade de cores, melhor fica a imagem, entretanto, maior também será o arquivo digitalizado. As opções possíveis são: • Colorido; • Tons de cinza; • Preto e branco.	A resolução está relacionada ao quanto de informação se tem em uma determinada área. Quando maior a resolução melhor a qualidade. Mas muita resolução também produz um arquivo muito grande. As opções possíveis geralmente vão de 50 a 10.000 DPI, mas os valores que mais utilizamos são os abaixo: • 96 DPI; • 150 DPI; • 300 DPI; • 4800 ou 9600 DPI.

Colorido
O colorido é ideal para digitalizar imagens coloridas como documentos contendo gráficos ou fotos. Nesse formato o arquivo fica maior do que nos formatos seguintes.

Tons de cinza
É uma opção intermediária para digitalizar documentos. Cores são transformadas em tons de cinza e o resultado é um arquivo menor que a opção anterior. Ideal para quando se tem fotos e imagens cuja informação sobre cores não é importante para o registro do documento. Este é o padrão mais utilizado para documentos SIGAD.

Preto e branco
Preto e branco é o formato mais simples, resultado ruim para imagens ou gráficos. Formato ideal para documentos do tipo texto puro. É o que gera o menor arquivo dos três.

96 DPI
Produz o menor arquivo das opções citadas, produz um bom resultado para arquivos que são apresentados na tela, mas um resultado ruim para arquivos impressos.

150 DPI
Produz um arquivo de tamanho intermediário entre as opções citadas, tem boa qualidade para apresentação em tela e qualidade moderada quanto impresso.

300 DPI
Produz arquivos grandes, excelente qualidade para documentos apresentados em tela e ótima qualidade para documentos impressos. Esse é o padrão mais utilizado para documentos SIGAD.

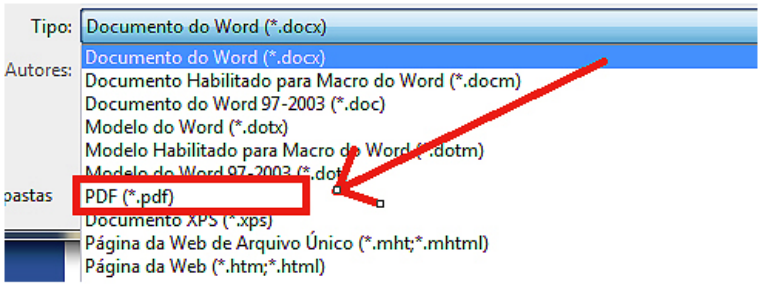
4800 ou 9600 DPI
Produz arquivos gigantes, utilizado raríssimamente, somente quando se precisa obter o máximo de informação. Nunca é utilizado para documentos do tipo texto. Só é útil para digitalização de fotos e imagens.

29

5.4 - Convertendo documentos eletrônicos em Word ou Excel para o formato PDF.

É possível você criar um documento no Ms Word, imprimir o documento e depois digitalizá-lo por meio de um scanner. Entretanto essa não é a melhor forma, pois o arquivo final perderá um pouco da qualidade em relação ao arquivo original.

A melhor forma de converter um arquivo Word, Excel, PowerPoint, E-mail e outros no formato PDF é utilizar a funcionalidade “salvar como” do software que criou o arquivo original. Dessa forma, o documento final será digitalmente fidedigno ao arquivo original, sem nenhuma perda criada durante a impressão ou a digitalização. Os pacotes de escritório OpenOffice e LibreOffice também possuem essa funcionalidade.



Utilize sempre a opção “Salvar Como” para criar arquivos PDF de alta qualidade.

30

5.5 - Convertendo qualquer arquivo do computador ou da Web para o formato PDF

Hoje em dia é possível converter qualquer informação que possa ser impressa em arquivo PDF. Você pode acessar uma página Web, por exemplo, e convertê-la em PDF. Ou pegar a imagem da tela do seu computador e também convertê-la em PDF.

Há softwares que fazem isso de uma forma muito inteligente e prática: eles criam uma impressora virtual que ao enviar o documento para essa impressora, ao invés de o documento sair impresso, ele é salvo em um documento PDF no seu computador. Saiba+

Tela do PDF Creator no momento que você enviar um documento para ser impresso nele.

Saiba+
Caso você queira testar essa funcionalidade, baixe e instale um dos aplicativos sugeridos abaixo:

- PDF Creator - www.pdfforge.org/pdfcreator.
- Cute PDF - www.cutepdf.com.

31

RESUMO

Neste módulo, aprendemos:

a. Que arquivos de texto formatado surgiram na era PC no final dos anos 1970, com os programas WordStar e WordPerfect.

b. Que o RTF foi o primeiro padrão a aceitar formatação complexa de texto, fontes, tabelas e imagens.
c. Que o RTF é um padrão recomendável para órgãos públicos trocarem informações de documentos de texto.
d. Que o RTF utiliza texto plano com delimitadores, o que torna a leitura do arquivo original possível e fácil de interpretar.
e. Que o DOC surgiu na década de 80 como o primeiro padrão totalmente gráfico e popularizado entre as pessoas comuns (além das empresas).
f. Que o padrão DOCX surgiu para permitir o registro de informações em formato XML e pode ser compactado sem perder qualidade (zipado).
g. Que a melhor forma de manipular arquivos DOC ou DOCX é por meio de componentes Office.
h. Que o padrão ODF é um formato aberto e internacional para documentos do tipo escritório (Office), utilizado por vários softwares livres. O ODF é um padrão em XML compactado. Os principais tipos de documentos ODF são: ODT (para textos), ODS (para planilhas) e ODP (para apresentações).
i. Que documentos do tipo planilha apesar de também ser facilmente manipuláveis via programação, na prática, existem poucas necessidades para esse tipo de manipulação.
j. Que documentos digitalizados são aqueles que convertem uma informação física em um arquivo digital.
k. Que podemos digitalizar documentos por meio de scanners e também máquinas fotográficas.
l. Que os principais tipos de arquivo digitalizados são PDF, TIFF e JPG, sendo o PDF o padrão mais bem aceito internacionalmente.