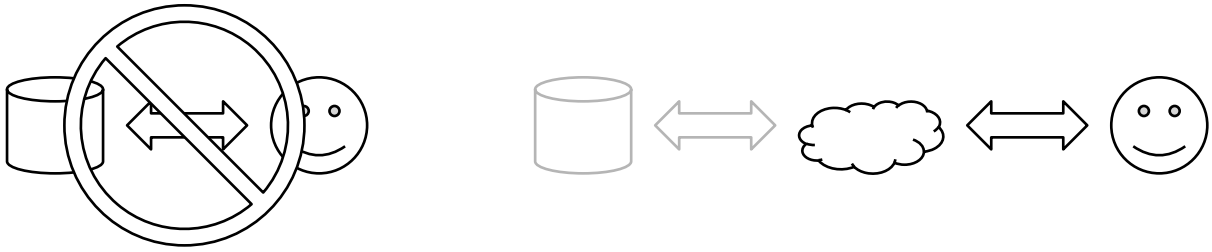
1. O objetivo desta lição é abordar intuitivamente formatos de dados na computação. Dois grandes grupos de formatos são visados: binários e textuais.

2. Um formato é um código, isto é, uma convenção. Em computação, os códigos versam sobre signos. Portanto, um código em computação corresponde a uma tabela que associa significantes e significados. Os significantes são estados da máquina, ou seja, a condução de transistores, a direção de campos magnéticos em discos, a posição de chaves, botões, etc. Os significados variam de acordo com os códigos: números, no caso dos códigos numéricos, desenho de caracteres, no caso dos códigos de caracteres, ações de um ambiente de execução (por exemplo, o processador e os demais componentes do hardware), no caso dos códigos de programas.

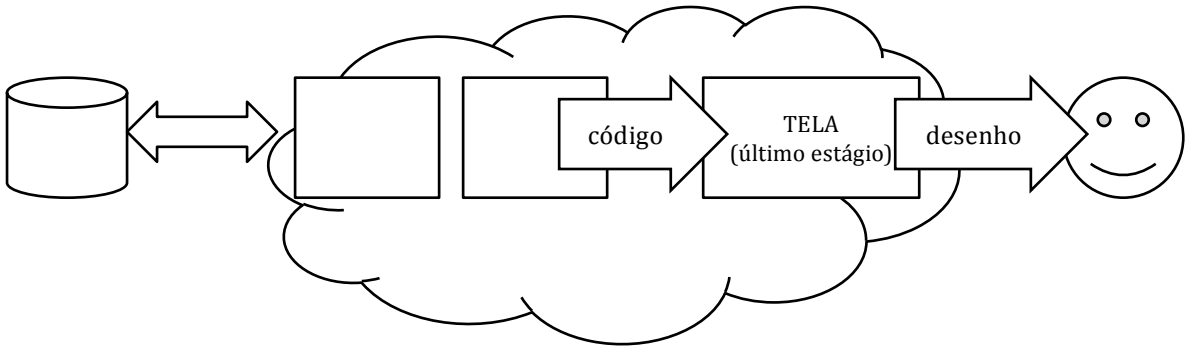
Há diversos códigos numéricos. Números inteiros e números reais não podem ser representados por um mesmo código. No caso dos números inteiros, em particular, se não for preciso representar números negativos, pode-se usar um código diferente.

3. Houve uma época em que as pessoas tinham contato direto com os mecanismos internos de um computador. Porém, já faz muito tempo que não mais é assim. Todo acesso aos estados da máquina é feito por intermédio de programas e interfaces (tela, etc.). São como instrumentos de visualização, "dadoscópios". Programas e interfaces sempre interpretam os dados de alguma forma quando os apresentam.

Como não há acesso direto aos estados da máquina, não existe uma forma mais correta de representá-los. Sabe-se que os computadores são binários, mas, exceto por questões de desempenho em certas operações, nada obriga que o formato binário seja o formato correto de visualização de um computador. O computador poderia ser ternário ou decimal. Quem determina o formato mais conveniente de visualização dos estados da máquina é o programa e a interface usados para acessá-los. O formato binário tem sua aplicação mesmo quando não se usam computadores, devido à sua alta compactação de informação.



4. Toda interpretação feita por programas e interfaces termina com uma decodificação de caracteres. Em outras palavras, tudo que se *lê* na tela são caracteres, pois a tela "desenha" caracteres a partir de códigos de caracteres.



Um arquivo no formato texto, isto é, um arquivo codificado segundo um código de caracteres, requer quase nenhum processamento para ser visualizado, pois a tela naturalmente mostra o desenho do caracter de cada código contido em cada posição do arquivo (interpretação).

5. Já foi dito que não existe acesso direto aos estados internos de um computador, e, portanto, todo acesso é feito mediante um intermediário que interpreta e exibe os dados de alguma forma. O mais próximo que se chega de uma visualização neutra, isto é, o mais sem interpretação possível, é através de uma representação dos dados como números inteiros sem sinal. Esse formato é mais conhecido como formato binário.

O mais correto seria chamá-lo de formato numérico. A base mediante a qual o número é representado não muda o número. O formato hexadecimal tem a vantagem de ser mais compacto e ser facilmente convertido para o formato binário mentalmente. O formato octal historicamente foi muito usado.

Antes de ser apresentado na tela o número precisa ser convertido em uma sequência de códigos de caracteres mediante os quais, como já foi discutido, a tela realiza o desenho dos algarismos do número. O dado 129, por exemplo, é apresentado em representação hexadecimal como 81. Em primeiro lugar, é preciso chegar ao valor 8. Ele corresponde à quantidade de "dezesseizenas" contidas em 129 (quociente da divisão inteira de 129 por 16). O código de caracter ASCII dos 10 algarismos indo-arábicos, começando pelo zero, é facilmente obtido somando-se 48 ao número (zero vale 48, um vale 49, dois vale 50, e, assim, por diante). Os algarismos hexadecimais de A a F são obtidos somando-se o valor 55 (dez vale 65, onze vale 66, etc.). Apresentar o valor 56 (= 48 + 8) à tela faz com que o desenho '8' seja mostrado. O mesmo processo é feito com o resto da divisão de 129 por 16, que é o segundo algarismo, o 1. É mais trabalhoso apresentar os dados na tela segundo o formato binário (numérico).

Dados costumam ser visualizados em blocos. Abaixo, vê-se um trecho de um *dump* de dados, que tanto poderia ser de um arquivo quanto de alguma parte de memória. A primeira coluna indica o endereço do primeiro byte de cada linha, em hexadecimal. Ou seja, o dado `ca` está na posição 0; `fe`, na posição 1; `ba`, na posição 2, e, assim, sucessivamente.

```
0000000 ca fe ba be 00 00 00 34 00 1c 0a 00 06 00 0f 03
0000010 00 00 cc 15 09 00 10 00 11 0a 00 12 00 13 07 00
0000020 14 07 00 15 01 00 06 3c 69 6e 69 74 3e 01 00 03
0000030 28 29 56 01 00 04 43 6f 64 65 01 00 0f 4c 69 6e
```

Essa é a forma mais próxima de visualização dos dados "como eles são", isto é, sem uma interpretação considerável.

6. Existem arquivos que só contém texto, por exemplo, as instruções de instalação em uma distribuição de software. Existem arquivos que só contém dados, por exemplo, os dados das temperaturas médias em Rio Claro nos últimos 360 dias. No entanto, há arquivos mistos, com trechos de texto inseridos entre dados binários. Para esses casos, a melhor forma de visualização é interpretar os mesmos dados segundo dois formatos ao mesmo tempo.

```
00000000  ca fe ba be 00 00 00 34  00 1c 0a 00 06 00 0f 03  |.......4........|
00000010  00 00 cc 15 09 00 10 00  11 0a 00 12 00 13 07 00  |................|
00000020  14 07 00 15 01 00 06 3c  69 6e 69 74 3e 01 00 03  |.......<init>...|
00000030  28 29 56 01 00 04 43 6f  64 65 01 00 0f 4c 69 6e  |()V...Code...Lin|
00000040  65 4e 75 6d 62 65 72 54  61 62 6c 65 01 00 04 6d  |eNumberTable...m|
00000050  61 69 6e 01 00 16 28 5b  4c 6a 61 76 61 2f 6c 61  |ain...([Ljava/la|
```

O *dump* acima, além da interpretação binária usual, inclui, na parte direita, a interpretação textual dos mesmos dados. Por exemplo, o dado na posição 52 (em hexadecimal) é mostrado na forma hexadecimal 6e e como o caracter 'n'.

7. Prática de laboratório

Recursos necessários:

- Sistema operacional Unix.
- O aplicativo `cat` (comumente instalado por padrão), para exibir arquivos de texto na tela.
- O aplicativo `hexdump` ou similar (instalar), para exibir *dumps*.
- Um arquivo texto, por exemplo, `/etc/passwd`
- Um arquivo binário, por exemplo, `/bin/ls`

Execute na linha de comando os seguintes casos e observe os efeitos:

`cat /etc/passwd`	`hexdump –C /etc/passwd`
`cat /bin/ls`	`hexdump –C /bin/ls`

8. Exercícios

(8.1) Protocolos de comunicação podem ser do tipo binário ou texto. Quais as vantagens de cada um? Como seria transmitir dados binários no formato texto? Como seria recebê-los?

(8.2) Para entender a necessidade de uma etapa a mais na exibição em formato binário, imagine que não existisse o procedimento `printf` em C, apenas `putchar`. Imprima o valor de uma variável inteira no formato decimal usando exclusivamente o procedimento `putchar`.

(8.3) Acesse uma página web em português. Modifique a codificação de texto do navegador entre as opções Latino-1 e Unicode. Mostre o efeito da alteração em alguma palavra acentuada. Explique por que o efeito ocorre.

9. Atividade (ver AVA)

Implemente, em Java, um programa de *dump* de arquivos. O formato exibido deve ser misto (hexadecimal + texto).