Introdução a Ciências de Computação II – Trabalho 1

1. Especificação e Descrição do Problema

Considere um arquivo de áudio gravado em formato binário. Esse arquivo contém amplitudes do áudio gravadas em blocos de bytes. As amplitudes são resultantes de variações no diafragma do microfone. Ao gravar o áudio, conforme falamos no microfone, há variações no diafragma, que são codificadas em números a cada instante, e gravados em um arquivo binário.

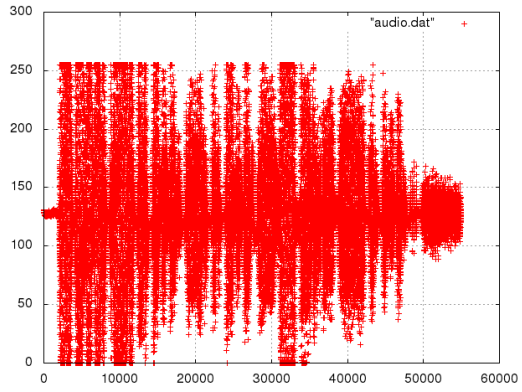
O valor da pressão de nossa voz sobre o diafragma do microfone é medido por um *unsigned byte*. Dessa maneira há 8 bits para representar a situação do diafragma em dado instante de tempo. Cada valor de amplitude varia de 0 até 255, pois ele é representado por um unsigned byte (ou seja, 1 byte sem sinal), em que há 8 bits que podem ser usados: logo $2^8 = 256$ possíveis valores. Como começa em 0 vai até 255.

O valor 127 (valor central) significa que o diafragma está em repouso, ou seja, ninguém está falando. Valores abaixo de 127 e acima de 127 significam variações do diafragma impostas por nossa voz.

Ao ler esse arquivo binário, **um byte por vez**, você teria valores como:

127 128 180 182 187 190 183 172 171 169 127 128

Ao construir um gráfico (plotar) esse arquivo com inteiros no formato texto (no Linux, pode-se usar o comando gnuplot para plotar) temos algo na forma:



Em que temos o tempo no eixo x e a amplitude do diafragma (1 byte) no eixo y. Como o arquivo produzido foi gravado com 8000 Hertz, há 8000 observações do diafragma por segundo de gravação. Esse número define a amostragem no tempo. Na Figura acima há cerca de 55000 observações no total. Dividindo as 55000 observações por 8000 (amostragem no tempo) é possível saber que esse arquivo representa 6,8 segundos de gravação.

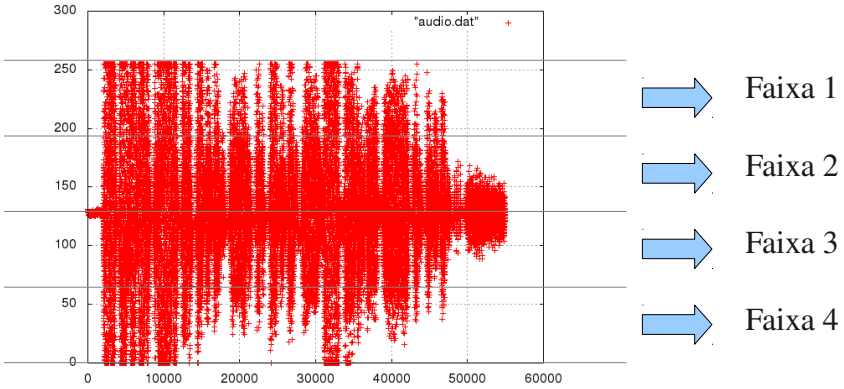
A sua tarefa nesse trabalho consiste em **ler arquivos de áudio binários** e t**ransformar os dados em N faixas de amplitudes médias**. Por exemplo, considere que o arquivo contém os seguintes valores de amplitude:

127 128 180 182 187 190 183 172 171 169 127 128

Crie N faixas de amplitudes médias. Para isso você deve dividir o intervalo de possíveis valores em N trechos. Veja um exemplo abaixo:

Seja N = 4
O intervalo original é de 0 até 255
Dividindo em 4 faixas: 256 / 4 = 64 valores por faixa

Agora você deve calcular a média dos valores de amplitude dentro de cada faixa:



Calcule a média dos valores por faixa. Suponha que dentro da Faixa 1 existam somente os seguintes valores:

250 253 255 198 198 255 253 253

Calculando a média temos: 239,375

Para que o valor seja inteiro, pegamos apenas a parte inteira da média, obtendo: **239,00**. Assim faremos para cada faixa.

Em seguida, iremos percorrer o arquivo todo e substituir cada valor pela média de sua faixa. Por exemplo, todas as vezes que encontrarmos o valor 198, por exemplo, iremos substituir pela parte inteira da média de sua faixa, ou seja, iremos substituir por 239, que é a média dessa faixa. Se encontrarmos 253, 255 e outros valores ainda na primeira faixa, também iremos substituir pela parte inteira da média dessa faixa, ou seja, 239.

Sendo assim os valores anteriores seriam transformados em:

239 239 239 239 239 239 239 239

Dessa maneira, o arquivo original de áudio será transformado em um novo arquivo binário de saída que

não tem mais os mesmos valores de amplitudes, mas sim valores médios de amplitude por faixa. Pode-se tocar esse arquivo de saída binário (no Linux basta usar o comando 'aplay -t raw saida.raw') e ainda ouvir o áudio original com pequenas distorções.

Tente depois compactar o arquivo binário original e esse novo arquivo de saída. Veja que o arquivo de saída compactado é menor. Isso ocorre porque ao substituirmos as amplitudes do diafragma pelas médias das faixas, temos muitos valores repetidos dentro do novo arquivo de saída. Assim, qualquer compactador como Gzip, Bzip2, Winzip e Zip é capaz de compactar muito mais o arquivo, pois eles buscam por repetições, ou redundâncias, e tentam evitá-las.

Essa é uma das maneiras mais simples de compactação de dados. Apesar de reduzir bastante a quantidade de dados, ela ainda permite ouvir e interpretar o áudio, desde que haja um número mínimo de faixas.

2. Implementação, entrada e saída

Você deverá implementar um programa em linguagem C que resolva o problema descrito na seção 1.

Esse programa deverá ser capaz de receber como entrada:

1. o nome de um arquivo de áudio no formato **.raw**;
2. o número N de faixas a serem computadas para o novo arquivo.

O programa deverá gerar como saída, caso as entradas estejam corretas:

1. um novo arquivo binário de áudio no formato **.raw**, gravado com as faixas médias conforme o parâmetro N definido na entrada;
2. exibir na tela do computador:
   1. o número (inteiro) total de observações presentes no arquivo (e lidas pelo programa);
   2. o número (inteiro) de observações que recaíram em cada uma das N faixas;
   3. o total em segundos do arquivo de áudio (considerando que todo arquivo de áudio foi gravado usando 8000 Hertz), no formato: **999.9**.

Exemplo:

./programa audio01.raw 6

A saída possui o seguinte formato:

26500**\n**
7200_2500_7000_9000_500_300**\n**
3.3**\n**

Onde **\n** significa uma quebra de linha e _ significa um espaço em branco

Para criar arquivos de áudio:

Dois arquivos já são fornecidos junto com o trabalho. No entanto, se o aluno deseja criar arquivos de áudio em Linux, basta utilizar os comandos a seguir:

arecord -t raw novoarquivo.raw

E para tocar um arquivo de áudio no Linux basta usar:

aplay -t raw novoarquivo.raw